PingDirectory

Contents

Release Notes

Directory Server Release Notes

PingDirectory 8.2.0.8 release notes

The release notes for the 8.2.0.8 release of the PingDirectory server.

Critical fixes

This release of the PingDirectory server addresses critical issues from earlier versions. Update all affected servers appropriately.

No critical issues have been identified.

Resolved issues

The following issues have been resolved with this release of the PingDirectory server.

Ticket ID	Description
DS-45164	Updated Jetty Server to version 9.4.44.
DS-45813	Updated PingDirectory products to use Kafka v2.8.1.
DS-45863	Resolved an issue where SCIM POST requests that violated a unique attribute constraint received an error response with status 400 (Bad Request) instead of 409 (Conflict).
DS-46017	Resolved an issue with slow response time on PingDirectory servers configured with a large number (10,000 or more) of virtual static groups.

PingDirectory Server 8.2.0.7 release notes

The release notes for the 8.2.0.7 release of PingDirectory Server.

Critical fixes

This release of PingDirectory Server addresses critical issues from earlier versions. Update all affected servers appropriately.

- Fixed an issue where secret keys under `cn=Topology,cn=config` could be lost when removing a server from the topology. When a server is removed via the `dsreplication disable` or **`remove-defunct-server`** tools, its secret keys will now be distributed among the remaining members of the topology. The keys from the rest of the topology will also be copied to the server being removed.

  The cipher secret keys in the topology that are affected by this change are used by reversible password storage schemes (except for AES256, which uses the encryption settings database). If you are using a

reversible password storage scheme other than AES256, prior to this fix, you could lose access to keys that had been used for reversible password encryption when removing servers from the topology.

> ⓘ **Note:**
>
> Since this change only applies to the most recent version of **`remove-defunct-server`** and `dsreplication disable`, if you are removing a server from a multi-version topology, you should run that tool from the most recent version. In the past **`dsreplication`** and **`remove-defunct-server`** could only be run from an older version, but now in the case of removing a server from the topology, they should be run from the most recent version in the topology. If you run the tool from an older server, it will not include this fix, and you might lose access to secret keys from servers that are removed from the topology.

- Fixed in: 8.2.0.7
- Introduced in: 7.0.0.0
- Support identifiers: DS-44591

Resolved issues

The following issues have been resolved with this release of the Directory Server.

Ticket ID	Description
DS-42961	To help with replication backlog analysis the replication summary monitor will now include a `replica-partial-backlog` attribute that will show how each origin replica contributes partial backlog with the `per-origin-replication-backlog` property. The `replica-partial-backlog` attribute will also show the change numbers used for the calculation.
DS-43959, DS-44924	Added new global configuration properties that can be used to impose limits on the maximum number of attributes that can be present in an add request and the maximum number of modifications in a **`modify`** request, which can be used to avoid potential denial of service attacks. Both limits are set to 1000 by default, which is likely to be adequate for all legitimate use cases, and neither property affects the number of values that may be provided for an attribute.

Ticket ID	Description
DS-44591	Fixed an issue where secret keys under `cn=Topology,cn=config` could be lost when removing a server from the topology. When a server is removed via the `dsreplication disable` or **`remove-defunct-server`** tools, its secret keys will now be distributed among the remaining members of the topology. The keys from the rest of the topology will also be copied to the server being removed. The cipher secret keys in the topology that are affected by this change are used by reversible password storage schemes (except for AES256, which uses the encryption settings database). If you are using a reversible password storage scheme other than AES256, prior to this fix, you could lose access to keys that had been used for reversible password encryption when removing servers from the topology. ⓘ **Note:** Since this change only applies to the most recent version of **`remove-defunct-server`** and `dsreplication disable`, if you are removing a server from a multi-version topology, you should run that tool from the most recent version. In the past **`dsreplication`** and **`remove-defunct-server`** could only be run from an older version, but now in the case of removing a server from the topology, they should be run from the most recent version in the topology. If you run the tool from an older server, it will not include this fix, and you might lose access to secret keys from servers that are removed from the topology.
DS-44892	Addressed a connection error in **`remove-defunct-server`** when the tool tried to migrate secret keys on a single-instance topology (i.e., a server that is not part of a replication topology). The tool now only moves secret keys if the server is part of a topology.
DS-45032	Addressed an issue that caused **`remove-defunct-server`** to hang when performing replication artifact cleanup in non-interactive mode.
DS-45115	Fixed a Ping Directory Server performance issue involving high CPU usage when writing LDAP data to certain clients using TLSv1.3 connection security.

Ticket ID	Description
DS-45124	Removed `-XX:RefDiscoveryPolicy=1` from the default start-server Java arguments. In rare cases, this argument was related to segmentation faults in the JVM, especially when used with the G1 garbage collector.
DS-45154	Fixed an issue where a server with a newly initialized database (through `dsreplication initialize`) could go into lockdown mode and report that the server may have missed one or more update(s). This generally occurred only if the initialized server was restarted right after initialization completed.
DS-45190	Added support for the use of JDKs obtained through BellSoft.
DS-45449	Updated the server to create the `esTokenizer.ping` file if it does not exist for a backend containing encrypted data. This file may be needed to open the database environment for a backend containing encrypted indexes, but it would not have been automatically created when upgrading from a pre-7.0 server to a later version with support for encrypted indexes.
DS-45654	Resolved an issue where SCIM POST requests that violated a unique attribute constraint got an internal error instead of the expected SCIM error response.

PingDirectory Server 8.2.0.6 release notes

Critical fixes

This release of PingDirectory Server addresses critical issues from earlier versions. Update all affected servers appropriately.

No critical issues have been identified.

Resolved issues

The following issues have been resolved with this release of the Directory Server.

Ticket ID	Description
DS-43576	Added the ability to retry operation failures from replication if the failures are likely due to dependent writes being played out of order. This issue only affected environments that were sending writes to different servers, and also were not able to use the appropriate level of replication assurance. To enable this setting, update the `on-replay-failure-wait-for-dependent-ops-timeout` configuration property on a replication domain.

Ticket ID	Description
DS-43632	Fixed an issue where the `format` field is omitted from the list of operational attribute schemas in the Directory REST API.
DS-43898	Addressed an issue where the backend would store an incorrect count for a database key that had exceeded the index entry limit. This would only happen for composite indexes when the server started up after not being shut down cleanly. Normal search processing was unaffected, but the matching entry count request control processing would return incorrect results.
DS-43999	Updated the server to allow the operation purpose control to be used for operations that are part of an LDAP transaction.
DS-44259	Fixed an issue that could prevent password policy state changes made through the `ds-pwp-modifiable-state-json` operational attribute from being properly replicated.

PingDirectory Server 8.2.0.5 release notes

Critical fixes

This release of the Directory Server addresses critical issues from earlier versions. Update all affected servers appropriately.

No critical issues have been identified.

Resolved issues

The following issues have been resolved with this release of the Directory Server.

Ticket ID	Description
DS-44025	Fixed an issue where the server was incorrectly displaying an `Unknown vendor` warning when using JDKs obtained on Red Hat and Ubuntu systems.
DS-44204	Fixed an issue in which the Directory Server could incorrectly allow requests to be processed with an alternate authorization identity (for example, using the proxied authorization control, or if the requests pass through a Directory Proxy Server) whose account is in a `must change password` state. The server will now only permit the operation if it attempts to set a new password for the target user.

PingDirectory Server 8.2.0.3 release notes

Critical Fixes

This release of PingDirectory Server addresses critical issues from earlier versions. Update all affected servers appropriately.

No critical issues have been identified.

Resolved Issues

The following issues have been resolved with this release of PingDirectory Server.

Ticket ID	Description
DS-43926	Significantly improved the performance of delete and modify operations that require removing IDs from very large composite index ID sets.

PingDirectory Server 8.2.0.2 release notes

Critical Fixes

This release of the PingDirectory Server addresses critical issues from earlier versions. Update all affected servers appropriately.

No critical issues have been identified

Resolved Issues

The following issues have been resolved with this release of the PingDirectory Server:

Ticket ID	Description
DS-43224	Made a generic OpenID Connect ID token validator available. This change allows single sign-on to the Administrative Console with OIDC providers other than just PingOne.
DS-43838	Updated the server's support for the uniqueness request control so that if a conflict prevention details entry is requested, the server may be able to use a minimal version of the proposed target entry instead of a full copy. This can help improve performance when using this option.
DS-43941	You can now specify that the Administrative Console use a custom truststore when evaluating OIDC provider certificates by using the oidc-trust-store-file and oidc-trust-store-type settings. Also, you can set the console to skip hostname and/or certification verification through the oidc-strict-hostname-verification and oidc-trust-all configuration settings.

PingDirectory Server 8.2.0.1 Release Notes

Critical Fixes

Resolved Issues

This release of the PingDirectory Server addresses critical issues from earlier versions. Update all affected servers appropriately.

No critical issues have been identified

The following issues have been resolved with this release of the PingDirectory Server:

Ticket ID	Description
DS-43024	Updated the purge expired data plugin to support deleting entries with multiple concurrent threads. Deletes will still be performed in a single thread by default, and this is the recommended configuration for most deployments, but the value of the num-delete-threads configuration property may be increased if expired entries accumulate faster than a single thread can delete them.
DS-43745	Fix an issue in the server causing PingDirectory 8.2.0.0 to be incompatible with version 4.2.0 of the Delegated Admin app, which prevented upgrade to Delegated Admin version 4.4.0 without some downtime.
DS-43770	Fixed an issue that could prevent pre-operation add plugins created with the Server SDK from accessing some changes to the content of the add operation. For example, a Server SDK pre-operation add plugin could not access an attribute created by a composed attribute plugin that had been invoked before the Server SDK plugin.

PingDirectory Server 8.2 release notes

Enhancements

These are new features for this release of PingDirectory Server:

- To improve our support for containerization, PingDirectory Server can now determine that it is running in a container and stores this state in the isDocker property within the System Information monitor (cn=System Information, cn=monitor).
- The Administrative Console now supports using OpenID Connect for admin SSO, allowing you to set up the PingOne administration console to have one-click SSO access without typing a password.
- To ensure that PingDirectory Server is protected against online password guessing attacks, PingDirectory Server can now inject a delay before sending failure responses back to the client for basic HTTP authentication failures.
- The Dictionary password validator includes approximately 10,000 more entries in the dictionary file and can now check for capitalization, reversed passwords, leading or trailing non-alphabetical characters and ignore diacritical marks.

- A new operational attribute has been defined to contain information about recent successful and failed bind attempts. Also, you can configure the server to maintain recent login history such as the number of records of successful and/or failed authentication attempts.
- The **`setup`** command can now create a populated `config/tools.properties` file from the properties set either on the command or input using interactive mode. There are several modes for adding the bind password to the file. By default, no properties file is created unless the --populateToolPropertiesFile argument is used (non-interactive mode) or the option to create the file is specified while running setup.
- The **`validate-ldap-schema`** tool has been added allowing for administrators to verify any LDIF file used to create schema and/or entries and report any potential issues they may contain.
- Typically, passwords are only validated when they are set, changed or reset for an entry. However, this might lead to a weak password since these evens are infrequent. PingDirectory Server now supports the ability to periodically run password validation on bind requests. By default, this is not enabled but can be added to any password policy by setting both the bind-password-validator and the minimum-bind-password-validation-frequency properties. If, at the time of the bind, the password does not meet the specified password policies, the bind-password-validation-failure-action property specifies what action to take like requiring a password change.
- Customers want a better way to perform a sorted search on a large dataset using SCIM 2.0. In this version of PingDirectory Server, SCIM searches with the proper search parameters can be the returned in pages. The PingDirectory Server needs to be configured with Virtual List View (VLV) indexes for the desired SCIM Resource Type.
- You can now configure password policies to keep a record of recent login history, successes, and failures, per account. The possible configuration options include tracking every success and failure or collapsing to a single event per day.
- To improve logging and monitoring, administrators can now configure the Syslog, Audit Log, HTTP Operation Log, and the Sync Log to produce output in the JSON format so other tools can more easily consume the information. These log publishers are disabled by default.

Upgrade considerations

Upgrade considerations are no longer part of the release notes. That information is now in *Upgrade overview and considerations* on page 268.

Known issues and limitations

The following are known issues in the current version of PingDirectory Server:

When signing on to the Administrative Console, you must specify an LDAPS port in the **Server** field. For example, to use port 636, you would specify pingdirectory:636. If you do not specify a port, you get a `"Server unavailable"` message.

Resolved Issues

The following issues have been resolved with this release of PingDirectory Server:

Ticket ID	Description
DS-644, DS-42031	Added a new validate-ldap-schema tool that can be used to examine schema definitions in a set of LDIF files and report any issues that it detects.
DS-1766	Updated the ldap-diff tool to make it possible to control the column at which long lines should be wrapped. Previously, long lines were always wrapped at 80 characters. Now, no wrapping is performed by default, but the --wrapColumn argument can be used to enable wrapping with a specified maximum line length.
DS-5040	Improved the **`dsframework`** tool to support multivalued server properties.

Ticket ID	Description
DS-5143, DS-11035	Updated support for logging access and error log messages to a syslog server. While the server previously supported logging these messages to a syslog server (through the "syslog-based access log publisher" and "syslog-based error log publisher" logger implementations), these loggers used an older version of the syslog protocol (described in RFC 3164) and only offered support for communicating over UDP. These loggers are still available for legacy backward compatibility, but we now also offer new "syslog text access log publisher" and "syslog text error log publisher" implementations that use a newer version of the syslog protocol (syslog version 1, described in RFC 5424) and support communicating over UDP or the more reliable TCP. When using TCP, it is also possible to encrypt communication with TLS, and it is possible to configure multiple servers for better redundancy. These loggers use the same space-delimited text format as the former loggers. We also offer new "syslog JSON access log publisher" and "syslog JSON error log publisher" implementations that offer the same set of capabilities, but that format the message text as JSON objects, which can be more easily parsed by third-party software.
DS-7475, DS-7605, DS-11725, DS-37707, DS-40342, DS-41940	Made several improvements to the parallel-update tool, which can use multiple concurrent threads for improved performance when applying add, delete, modify, and modify DN changes read from an LDIF file. The enhancements include: ▪ Added support for several additional controls, including proxied authorization, manageDsaIT, ignore NO-USER-MODIFICATION, password update behavior, operation purpose, name with entryUUID, assured replication, replication repair, suppress operational attribute update, and suppress referential integrity updates. The tool also now supports specifying arbitrary controls for inclusion in add, bind, delete, modify, and modify DN requests. ▪ Made its communication more robust. The tool would previously establish connections only when it was first started, but it can now detect when a connection is no longer valid and can re-establish connections as needed to continue processing. Further, if an operation failed because it was attempted on an invalid connection, that operation can be automatically retried immediately on a newly established connection. ▪ Added support for failover directory servers. You can now provide the --hostname and --port arguments multiple times to specify information about multiple directory server instances. In the event that the first server listed is not available (or becomes unavailable in the middle of processing), it can automatically try establishing a connection to an alternative server to continue processing there. ▪ Improved the ability to determine whether all changes were processed successfully. Previously, the tool would always use an exit code of zero if it was able to attempt all of the changes read from the LDIF file, even if some of the changes were not successfully applied. That is still the default behavior, but a new --useFirstRejectResultCodeAsExitCode argument can be used to indicate that if any operations are rejected, the result code from the first rejected operation should be used as the exit code. ▪ Added support for encrypted LDIF files. If the LDIF file was encrypted with a key from the server's encryption settings database, then the tool will automatically attempt to retrieve the appropriate key. Otherwise, the new --encryptionPassphraseFile argument can be used to supply the encryption passphrase, or the passphrase can be interactively requested from the user. ▪ The tool is now parallel by default. Previously, if you did not specify a value for the --numThreads argument, it would use a single thread. It now defaults to using eight threads. ▪ The tool now provides a --followReferrals argument that allows it to automatically attempt to follow any referrals that are returned. ▪ The tool now provides a menu-driven interactive mode that can be used to provide values for all of the command-line arguments.

Ticket ID	Description
DS-10320, DS-12550, DS-12551, DS-12552, DS-42116, DS-42162, DS-42179, DS-42222, DS-42223, DS-42224, DS-42225, DS-42416, DS-42437	Added a config/sample-dsconfig-batch-files directory with set of well commented dsconfig batch files that may be useful in enabling or configuring a variety of features in the server.
DS-10775	Updated the dictionary password validator to support additional options: ▪ It can now ignore non-alphabetic characters that appear at the beginning or end of the password before checking the dictionary. ▪ It can strip characters of diacritical marks, including accents, cedillas, circumflexes, diaereses, tildes, and umlauts, before checking the dictionary. If this option is used, then any character with such a mark will be replaced with a base version of the character without that mark (for example, a lowercase letter n with a tilde over it would be replaced with just a lowercase letter n). ▪ You can define maps with information about character substitutions to use for checking alternative versions of the provided password. For example, if you indicate that "0" might map to "o", "1" or "!" might map to "i", "7" might map to "t", and "3" might map to "e", then the validator can reject a proposed password of "pr0h1b!73d" if the dictionary contains the word "prohibited". ▪ It can reject a proposed password if a value from the provided dictionary makes up more than a specified percentage of that password.
DS-11524, DS-41860, DS-42112	Added support for new administrative alert types: ▪ We have added a new admin alert account status notification handler, which can generate administrative alerts whenever an applicable account status notification is generated within the server. For example, this account status notification handler can be added to the root password policy to generate an alert whenever a root user's password is updated or their account is locked as a result of too many failed authentication attempts. A separate alert type has been defined for each account status notification type. ▪ We have added a new "privilege-assigned" administrative alert that can be raised whenever a new entry is added or an existing entry is updated to include one or more privileges. ▪ We have added a new "insecure-request-rejected" administrative alert that can be raised whenever the server rejects a request as a result of the reject-insecure-requests global configuration property.
DS-13853	Added support for the OAUTHBEARER SASL mechanism (as described in RFC 7628) to allow LDAP clients to authenticate with OAuth 2.0 bearer tokens.
DS-15123	Updated the password policy to add support for maintaining a history of recent successful and failed bind attempts for each account. A new "get recent login history" control can be used to obtain this history during a successful bind, and it can also be obtained from the ds-pwp-state-json virtual attribute or the password policy state extended operation.

Ticket ID	Description
DS-15848, DS-42360	Added support for invoking a specified set of password validators during bind operations. If the password used to authenticate fails to satisfy one or more of the configured validators, the bind attempt can be rejected, the user can be forced to change their password, or the server can generate an account status notification to take some alternative action (for example, notifying the end user or server administrators).
DS-15864	Replaced the ldappasswordmodify tool with a new version that offers more functionality, including support for additional controls, support for multiple password change methods (the password modify extended operation, a regular LDAP modify operation, or an Active Directory-specific modify operation), and the ability to generate the new password on the client.
DS-17422, DS-37387	Added a new "Modifiable Password Policy State" plugin with support for a new ds-pwp-modifiable-state-json operational attribute whose value is a JSON object with fields that reflect elements of the associated user's current password policy state. Authorized clients may replace the value of that attribute (with a modify request using LDAP or with a PUT or PATCH request using the Directory REST API) to update that user's password policy state. Elements of the password policy state that may be updated include: ▪ The time the user's password was last changed ▪ The user's account activation time ▪ The user's account expiration time ▪ The user's password expiration warned time ▪ Whether the account is administratively disabled ▪ Whether the account is locked as a result of too many failed authentication attempts ▪ Whether the user will be required to change their password before they will be allowed to perform any other operations
DS-17664	Fixed an issue that could cause the server to return the password expired control (as defined in draft-vchu-ldap-pwd-policy) in response to a bind request that included invalid credentials. The server will now only return that control if the provided credentials were valid. The password policy configuration has also been updated to make it possible to better specify when the server should return the password expired or password expiring control. Previously, the server would only return these controls if the bind request did not include the password policy request control (as defined in draft-behera-ldap-password-policy), as the response to that control may also indicate whether the user's password is expired or is about to expire. This is still the default behavior, but it is now possible to configure the server to return the password expired or password expiring control when appropriate, even if the password policy response control will also be returned. Alternatively, the server can be configured to never return the password expired or password expiring controls.
DS-17903	Updated setup to provide a --populateToolPropertiesFile argument that will allow it to populate the config/tools.properties file with default values for command-line tool arguments. If requested, properties will be provided for the server address, port, and communication security, and may also include a default bind DN and optionally a bind password. When running setup interactively, it will now prompt to determine which properties (if any) should be populated in the properties file.

Ticket ID	Description
DS-36088	Updated the crypto manager to make it possible to augment the set of enabled TLS cipher suites with specific suites to add to or remove from the default set of enabled suites. To enable one or more suites in addition to those in the default set, prefix the names of those suites with the "+" symbol. To disable one or more suites in the default set of enabled suites, prefix the names of those suites with the "-" symbol. This was already possible when configuring cipher suites for the LDAP and HTTP connection handlers, but it was not an option for the crypto manager.
DS-38110	Updated the System Information monitor with an "isDocker" attribute to identify if the server is running in a Docker container.
DS-38118, DS-42495	Made several updates related to the server's handling of data written to standard output and standard error: ▪ The server can now be configured to rotate the logs/server.out file once it reaches a given size, and it will retain a configurable number of those log files. By default, the server will rotate the file once it reaches 100 megabytes and will keep up to ten files. ▪ To better facilitate capturing log data in containerized environments, the server now supports writing JSON-formatted access and error log messages to the JVM's original standard output and error streams (which will be separate from the server.out file when the server is started with the --nodetach argument). ▪ It is now possible to prevent the server from logging messages during startup in non-JSON format. It is also possible to prevent messages about administrative alerts from being written to standard error, or to write those messages in JSON format. These options are especially useful when using JSON-based logging to the console in no-detach mode, as they can help ensure that everything written to standard output and standard error will be formatted as JSON objects.
DS-38816, DS-41995	Updated support for the uniqueness request control to provide a more reliable mechanism for preventing conflicts that arise from operations processed concurrently in the same or different servers. If indicated in the request control, a temporary conflict prevention details entry can be added to the server before searching for existing conflicts, and that entry can be detected during pre-commit processing for other operations with uniqueness request controls that attempt to make a conflicting change. The server has also been updated to make it possible to generate an administrative alert if a uniqueness conflict is detected during post-commit processing for the uniqueness request control. Even though the conflict cannot be prevented at this stage in processing, the alert can let administrators know about it as soon as it happens so they can take any appropriate corrective action.
DS-38868	Updated setup to create a second encryption settings definition if data encryption is enabled. It will continue to create a definition for 128-bit AES encryption for use as the preferred definition to preserve backward compatibility with existing servers in the topology, but it will now also generate a definition for 256-bit AES encryption. The 256-bit AES definition may become the preferred definition in a future release, but you can use it now by first ensuring that any existing instances are updated to contain the new definition (with the "encryption-settings export" and "encryption-settings import" commands) and then making it the preferred definition (with "encryption-settings set-preferred") in all instances.

Ticket ID	Description
DS-39257	The **`dsreplication enable`** subcommand options `--preferredID1` and `--preferredID2` allow you to specify a preferred value for new replicationIDs for the first and second server respectively. By default, replication IDs are randomly chosen. The lowest unused value that is greater than or equal to the value specified is used, subject to being even or odd as required. Replication server IDs must be even, and replicaIDs must be odd. Each value specified must be in the range [0, 32766] inclusive. If the maximum value is reached while searching, the search wraps to 0.
DS-39789	Updated the JVM memory usage monitor provider to fix an issue that could prevent the monitor from reporting the total amount of memory held by all memory consumers. Also, fixed an issue that could cause the memory-consumer attribute to use an incomplete message for consumers without a defined maximum size and added an additional memory-consumer-json attribute whose values are JSON objects with data that can be more easily extracted by automated processes.
DS-40650	Updated the collect-support-data tool to make it possible to specify how much data should be captured from the beginning and end of each log file to include in the support data archive. You can also specify the capture size when invoking the tool through an administrative task, recurring task, or extended operation.
DS-40828	Fixed an issue where some state associated with a JMX connection was not freed after the connection was closed. This led to a slow memory leak in servers that were monitored by an application that created a new JMX connection each polling interval.
DS-40967	Eliminated a misleading error message that could be logged at startup if the server was configured with one or more ACIs that only apply when using specific SASL mechanisms.
DS-41228	Missing changes will now be detected when the backend is reverted and there are insufficient changes in the changelog database. When in this particular missing changes state the local replica will not accept changes from the local replication server.
DS-41230	To assist with recovering from the split-brain state the "dsreplication initialize" command will have a new "--force" option that overrides the lockdown check.
DS-41350	Fixed an issue where disabling certain backends (such as 'alarms') caused an internal monitor to log unnecessary error messages every few seconds, about not being able to gather data from that backend. Note that deliberately disabling the 'alarms' backend is not recommended in normal operation, but may occur during backup/restore operations.
DS-41521	The SCIM 2 service on PingDirectory and PingDirectoryProxy now automatically generates a Swagger 2 specification document based on the server's SCIM 2 configuration. To view this document, go to https://<your-server>/api-docs in a web browser.
DS-41707	To help with multi-tenant and delegated admin ACI parameterization implement the ($attr.attrName) macro. In particular the ($attr.attrName) macro can be used to build bind DNs.
DS-41865	PingDirectory and PingDirectoryProxy can now perform SCIM 2 paged searches on result sets greater than the configured lookthrough limit. To perform such a search, you must first configure a virtual list view (VLV) index.

Ticket ID	Description
DS-41964	Fixed an issue with the manage-profile tool where files in a server profile's dsconfig/ directory without a ".dsconfig" extension could cause failures in manage-profile replace-profile when validating updated dsconfig files.
DS-41989	Fixed an issue that could result in duplicate column headers being produced by the Periodic Stats Logger, even when the header-prefix-per-column attribute was set to true.
DS-42045	Updated the Stats Collector Plugin with a new generate-collector-files configuration property. When using the plugin exclusively for providing metrics to one or more StatsD Monitoring Endpoints, set this property to false to prevent unnecessary I/O.
DS-42059, DS-42060	Updated setup to add options for improving communication security: ▪ Non-interactive setup now offers a --rejectInsecureRequests argument that will configure the server to reject any request received over a connection that is not encrypted with SSL or StartTLS. ▪ Non-interactive setup now offers a --rejectUnauthenticatedRequests argument that will configure the server to reject any request received over a connection that is not authenticated (or that is authenticated as the anonymous user). ▪ Interactive setup now allows you to configure the server with the LDAP connection handler disabled (which was already an option when using non-interactive setup), or enabled but only for communication encrypted with StartTLS. The --rejectInsecureRequests and --rejectUnauthenticatedRequests arguments can also be used with manage-profile by including them in the setup-arguments.txt file of the server profile.
DS-42061	Updated the interactive command-line tool framework to prefer establishing secure LDAP connections over insecure connections. Previously, when prompting for the information needed to establish a connection, the default option was to create an unencrypted LDAP connection. Now, tools will default to creating an SSL-encrypted connection if the server supports it, or to creating a StartTLS-encrypted connection if that is available but SSL is not. Tools will also default to using streamlined settings when establishing secure connections. Previously, they would always prompt about how to determine whether the server's certificate chain should be trusted. When using the streamlined settings, the tools will only prompt about certificates that cannot automatically be considered trusted using information in the JVM's default trust store, the server's default trust store (config/truststore), or the server's topology registry.
DS-42062	Updated the root password policy so that LDAP bind responses for root users and topology administrators will be delayed by one second after five consecutive failed authentication attempts.
DS-42063	Updated the "delay bind response" failure lockout action to provide an option to delay the response to bind requests initiated by non-LDAP clients (for example, when using HTTP basic authentication). This option is disabled by default because delaying the bind response for non-LDAP clients may require temporarily blocking the thread used to process the request, which could increase the risk of a denial-of-service attack. To help mitigate this risk, if you enable delayed bind responses for non-LDAP clients, we recommend that you also increase the number of request handler threads for all enabled HTTP connection handlers.

Ticket ID	Description
DS-42115	Updated the server's command-line tool framework to make it easier and more convenient to communicate with the server over a secure connection when no trust-related arguments are provided. Most non-interactive tools will now check the server's default trust store, the topology registry, and the JVM's default trust store to see if the presented certificate chain can be automatically trusted without the need to prompt the user. If the presented chain cannot be automatically trusted, the user may be interactively prompted to determine whether it should be trusted.
DS-42157, DS-43033	Updated the CRAM-MD5 and DIGEST-MD5 mechanism handlers so that they are no longer considered secure. Although the credentials are encoded in transit, their protection relies on the weak MD5 digest. Further, they require that user passwords be encoded in a reversible form so the server can retrieve them in the clear for use in authentication processing, which increases the risk that they will be exposed in a data breach. This primarily affects the ability to use these SASL mechanisms over an unencrypted connection for users that are required to authenticate in a secure manner (for example, if their password policy has require-secure-authentication set to true, or if their entry has a ds-auth-require-secure-authentication operational attribute with a value of true). These SASL mechanisms are still enabled by default for legacy backward compatibility purposes, but we discourage their use. To assist with that, we have also provided a sample dsconfig batch file that can be used to disable these SASL mechanism handlers.
DS-42164, DS-43015	Added a new AES256 password storage scheme that can encode passwords with 256-bit AES encryption. Although non-reversible schemes are recommended in most deployments, reversible schemes may be necessary under certain circumstances, and the new scheme provides support for stronger encryption and better integrity protection than the existing 128-bit AES scheme. The key used to encrypt passwords will be derived from an encryption settings definition. Clients with access to an AES256-encoded password and the passphrase used to encrypt it may obtain the original plaintext representation of the password. The UnboundID LDAP SDK for Java provides support for decoding and decrypting these passwords through the AES256EncodedPassword class, and the Javadoc for that class provides documentation that may be used to implement support those passwords in other programming languages.
DS-42199	Optimized some searches commonly used by the status tool. This should improve the performance of the tool in more complex or large-scale environments.
DS-42265	Upgrade to jetty 9.4
DS-42276	Fixed an issue where using the encryption-settings tool to import definitions with the set-preferred flag could result in none of the imported definitions being set as the preferred definition.
DS-42279	Updated the server to require a minimum key size of 2048 bits when negotiating a TLS cipher suite that uses ephemeral Diffie-Hellman key exchange.
DS-42298	Replaced the ldifsearch, ldifmodify, and ldif-diff command-line tools with more full-featured and robust implementations.
DS-42331	Replaced the ldapcompare tool with a new version that offers more functionality, including support for multiple compare assertions, following referrals, additional controls, and multiple output formats (including tab-delimited text, CSV, and JSON).

Ticket ID	Description
DS-42347	Updated the server to use /dev/urandom (on non-Windows systems where that path exists and is readable) instead of /dev/random as the primary source for secure random data. Attempts to read from /dev/random can block if the underlying system does not have sufficient entropy, which can have a severe adverse effect on performance. Reads from /dev/urandom will not block, and the data that it provides is no less secure than data from /dev/random in any way that matters for the server.
DS-42349, DS-43209, DS-43210, DS-43323, DS-43324	Added support for JSON-formatted audit loggers, which complement the existing file-based LDIF-formatted error logger. The JSON-formatted audit log messages provide a record of changes to data in the server and can be written to files on the local filesystem, written to the JVM's standard output or standard error stream, or sent to a syslog server. Added support for JSON-formatted HTTP operation loggers, which complement the existing file-based loggers using the W3C common log format and a proprietary space-delimited text format. The JSON-formatted HTTP operation log messages provide a record of interaction with HTTP clients and can be written to files on the local filesystem, written to the JVM's standard output or standard error stream, or sent to a syslog server. Fixed an issue that caused JSON-formatted loggers to use a timestamp format that was not strictly compliant with the ISO 8601 format described in RFC 3339. Timestamps incorrectly omitted the colon between the hour and minute components of the time zone offset.
DS-42362	Fixed an issue in which the server could incorrectly evaluate a matched values request control containing an extensible match filter that specified both an attribute type and a matching rule. The server incorrectly used the attribute type's equality matching rule instead of the matching rule specified in the filter.
DS-42373	Fixed an issue where unindexed LDAP searches filtered on virtual attributes sometimes omitted objectClass from the returned result.
DS-42381	Fixed an issue that could prevent the uninstaller from removing information about the instance from the topology registry.
DS-42405	Fixed an issue that could raise an internal error when trying to undelete a non-soft-deleted entry.
DS-42438	Fixed an issue that could cause the remove-defunct-server tool to not remove certain replication attributes when run with a topology json file.
DS-42504	Updated manage-profile replace-profile to set encryption settings definitions defined in the newer server profile as preferred in the encryption settings db.
DS-42505	Previously, even if LDAP SDK debug logging was enabled, messages would be suppressed while running dsreplication. Now if LDAP SDK debug logging is enabled, messages will get printed to the console while running dsreplication.
DS-42527	Fixed an issue that could cause an exception when creating a resource in SCIM 1.1 using certain types of DNTemplate.
DS-42547	Fixed an issue where manage-profile generate-profile would print "null" as the generated profile directory when writing to an existing directory.
DS-42609	Fixed an issue in which the Directory REST API could fail to decode certain credentials when using basic authentication.

Ticket ID	Description
DS-42609	Fixed an issue in which the Consent API could fail to decode certain credentials when using basic authentication.
DS-42632	Added support for creating or importing a key pair configuration object using an elliptic curve (EC) key algorithm. You can use this to designate the encryption key pair for a JWT access token validator that handles EC-encrypted access tokens.
DS-42634	The JWT Access Token Validator can now validate JWT access tokens signed using the elliptic curve digital signature algorithms ES256, ES384, and ES512.
DS-42635	The JWT Access Token Validator can now validate JWT access tokens encrypted using elliptic curve cryptographic algorithms. The following key encryption algorithms are now supported in addition to RSA-OAEP: ECDH-ES, ECDH-ES+A128KW, ECDH-ES+A192KW, and ECDH-ES +A256KW. To support best practices for JWT security, you must now also configure the JWT Access Token Validator with explicit allow lists for key encryption and content encryption algorithms. For backward compatibility, the key encryption allow list defaults to RSA-OAEP, while the content encryption allow list defaults to A128CBC-HS256, A192CBC-HS384, and A256CBC-HS512. We recommend setting both allow lists to the strict minimum set of algorithms needed by the Access Token Validator.
DS-42651	Updated the manage-profile replace-profile subcommand to better support updating the server's keystore and truststore files. When using the --generateSelfSignedCertificate argument in a server profile's setup-arguments.txt file, the server will maintain the original keystore and truststore files during replace-profile. Otherwise, replace-profile will use the keystore and truststore specified in the profile's setup-arguments.txt file.
DS-42667	Updated the server to set a unique cluster name when started for the first time.
DS-42669, DS-42748	Updated the online dsconfig step of the manage-profile replace-profile subcommand to support getting LDAP connection arguments from a tools.properties file on the server being updated. Fixed an issue where boolean LDAP connection arguments like --useSSL and --trustAll would cause manage-profile replace-profile to fail when applying dsconfig online.
DS-42673	Updated the manage-profile setup subcommand to fail if the start-server command has a non-zero exit code.
DS-42681, DS-42684	Performance statistics generated by the Sideband API can now be published by the Periodic Stats Logger. To enable this, use the "included-http-servlet-stat" property of the Periodic Stats Logger.
DS-42687	Upgrade to Jetty 9.4.30
DS-42720	Updated the ds-pwp-state-json virtual attribute provider to include information about the requirements that passwords must satisfy for adds, self password changes, administrative password resets, and binds.
DS-42740	Fixed an issue where the **`dsconfig ilst`** subcommand would not display requested properties.

Ticket ID	Description
DS-42749	To support best practices for JWT security, you must now configure the JWT Access Token Validator with an explicit list of the JWT signing algorithms that it accepts. For backward compatibility, this list defaults to the RSA signing algorithms RS256, RS384, and RS512, but we recommend setting this list to the strict minimum set of signing algorithms needed by the Access Token Validator.
DS-42751	Added new override-status-code and additional-response-contents attributes to the Availability State HTTP Servlet Extension. These new attributes can be used to customize the response code and JSON response body of the servlet.
DS-42756	Fixed a bug where backup retention attempted to purge old backups out of order, which lead to errors.
DS-42790	Fixed an issue where new replicas incorrectly went into lockdown mode after initialization. This issue would happen when trying to initialize a newly-added replica to a topology that had been created some time ago. This amount of time had to exceed the replication purge delay, which is 24 hours by default. Before this fix was introduced, you could get past this by running "leave-lockdown-mode" on the new replica, then re-running "dsreplication initialize" on it.
DS-42815	Added a remove-attribute-type-from-schema tool that can be used to safely remove an attribute type definition from the server schema. It will ensure that the attribute type is not in use, and it will clean up metadata references to that attribute type that could have previously required re-importing the data before the attribute type could actually be removed from the schema. The remove attribute type processing can also be requested programmatically through an administrative task.
DS-42834	Fixed an issue that could inappropriately allow HTTP-based clients (for example, those using SCIM, the Directory REST API, or the Consent API) to issue requests when the server is in lockdown mode.
DS-42850	Fixed a typo in the password-expiring template that caused "password_expiration_time_of_day" to be printed instead of the password expiration time.
DS-42861	Updated the manage-profile tool logs to include the duration of each step the tool takes. The new --verbose argument can also be used to display timing information in the tool's console output.
DS-42865	Added an aggregate identity mapper that uses Boolean AND or OR combinations of other identity mappers.
DS-42866	Updated the exact match and regular expression identity mappers to make it possible to only include entries matching a given filter.
DS-42872	Added a JSON-formatted stats logger to the server's default configuration. The stats logger is disabled by default.
DS-42886	Updated non-interactive setup (including manage-profile setup) to allow the password for the initial root user to be provided in pre-encoded form using the PBKDF2, SSHA256, SSHA384, or SSHA512 password storage scheme. This eliminates the need to have access to the clear-text password when setting up the server.

Ticket ID	Description
DS-42922	Fixed a bug where restoring an incremental backup could result in the server not being able to start.
DS-42926	Fixed an issue where Ping Directory products configured to run as Microsoft Windows services were sometimes unable to automatically restart following an unplanned reboot, due to errors reading a corrupted server status file.
DS-42939	The Administrative Console configuration settings have been updated to account for the new SSO functionality.
DS-42952	For Windows only, there can be a hang on start when global configuration property `startup-error-logger-output-location` is set to values that contain `standard-error`. For Windows only, `standard-error` values are silently mapped to equivalent `standard-output` values.
DS-42963	Updated the manage-profile generate-profile subcommand to ignore files larger than 100 megabytes when generating a server profile. Fixed an issue where many large files in the server root could cause the tool to run out of memory.
DS-43027	Added a new --adminPasswordFile argument to the manage-topology add-server command, to allow specifying the administrator password with a file rather than with the command line.
DS-43043	Fixed an issue where paged subtree searches posted to the Directory Rest API failed with error message: "Unable to decode the cookie in the simple paged results control value", whenever the search returned entries with DN length approaching or exceeding 127 characters.
DS-43047	Fixed an issue that could result in isMemberOf and isDirectMemberOf attributes not being updated appropriately when updating groups with multiple threads.
DS-43064	Fixed an issue that caused the server to return an objectClassViolation result code instead of the more appropriate attributeOrValueExists result code when trying to modify an entry to add an object class that already exists in that entry.

Ticket ID	Description
DS-43068	Added a new export-reversible-passwords tool that can be used to create an LDIF export containing the clear-text representations of reversibly encoded passwords. The export may optionally include non-password attributes from the entries, entries containing non-reversible passwords, and entries without passwords. This tool can be used to help rotate the keys used to encrypt passwords if the need arises. There are a number of safeguards in place to help ensure that this tool cannot be used inappropriately. These include: ▪ The tool invokes the export over LDAP and therefore requires that the server be online. It cannot be invoked with the server offline. ▪ The tool can only be invoked by a user with the permit-export-reversible-passwords privilege. This privilege is not granted to any users (even root users) by default. ▪ The tool can only be used if the server is configured with an instance of the "export reversible passwords" extended operation handler. The requester must also have access control permission to invoke this extended operation. ▪ The tool can only be run from the server system itself and the request must be received over the loopback interface. The export cannot be requested by a remote client. ▪ The request must be issued over a secure connection. ▪ The output file will be written to the server filesystem. It may only be written to a file that is beneath the instance root, and that file must not already exist (although its parent directory must exist). ▪ The exported file will be encrypted using the UnboundID LDAP SDK for Java's PassphraseEncryptedOutputStream, using a key generated from either a user-supplied passphrase or an encryption settings definition. This encrypted file may be directly imported by the import-ldif tool. If the contents of the file are needed, the LDAP SDK may be used to access its contents programmatically, or the encrypt-file tool may be used to decrypt it. ▪ The server will generate administrative alert notifications whenever it begins and ends the export process.
DS-43073, DS-43198	Added support for ID Token Validators, which validate the integrity and content of ID tokens issued by OpenID Connect providers. Use these validators with the OAuth Bearer SASL Mechanism Handler to enable single sign-on (SSO) for the Administrative Console using an OpenID Connect provider such as PingOne. Currently, only PingOne is supported for SSO.
DS-43074	Added three built-in identity mappers that you can use to look up administrative accounts stored in the server configuration: Root DN Users, Topology Admin Users, and All Admin Users.

Ticket ID	Description
DS-43288	Updated setup and the replace-certificate tool to improve the way we generate self-signed certificates and certificate signing requests to make them more palatable to clients. To reduce the frequency with which administrators had to replace self-signed certificates, we previously used a very long lifetime for self-signed certificates generated by setup or the replace-certificate tool. However, some clients (especially web browsers and other HTTP clients) have started more strenuously objecting to certificates to long lifetimes, so we now generate self-signed certificates with a one-year validity period. The inter-server certificate (which is used internally within the server and does not get exposed to normal clients) is still created with a twenty-year lifetime. Also, the replace-certificate tool's interactive mode has been updated to improve the process that it uses to obtain information to include in the subject DN and subject alternative name extension for self-signed certificates and certificate signing requests. The following changes have been made in accordance with CA/Browser Forum guidelines: ▪ When selecting the subject DN for the certificate, we listed a number of common attributes that may be used, including CN, OU, O, L, ST, and C. We previously indicated that CN attribute was recommended. We now also indicate that the O and C attributes are recommended as well. ▪ When obtaining the list of DNS names to include in the subject alternative name extension, we previously suggested all names that we could find associated with interfaces on the local system. In many cases, we now omit non-qualified names and names that are associated with loopback interfaces. We will also warn about any attempts to add unqualified or invalid names to the list. ▪ When obtaining the list of IP addresses to include in the subject alternative name extension, we previously suggested all addresses associated with all network interfaces on the system. We no longer suggest any IP addresses associated with loopback interfaces, and we no longer suggest any IP addresses associated in IANA-reserved ranges (for example, addresses reserved for private-use networks). The tool will now warn about attempts to add these addresses for inclusion in the subject alternative name extension.
DS-43305	Increased the maximum number of RDN components that a DN may have from 50 to 100.
DS-43376	Updated log publisher logic to reduce the amount of CPU that the server consumes when it is idle.
DS-43480	Updated the system information monitor provider to restrict the set of environment variables that may be included. Previously, the monitor entry included information about all defined environment variables, as that information can be useful for diagnostic purposes. However, some deployments may include credentials, secret keys, or other sensitive information in environment variables, and that should not be exposed in the monitor. The server will now only include values from a predefined set of environment variables that are expected to be the most useful for troubleshooting problems, and that are not expected to contain sensitive information.
DS-43517	Updated the jose4j library used for JWT signing and encryption to version 0.7.2.
DS-43651	The Security Guide is now available online at pingidentity.com. The guide has been removed from the server packaging.
DS-43666	Fixed an issue in which a server in lockdown mode could incorrectly allow an operation to be processed if a connection authenticated as a user with the lockdown-mode privilege issued a request with an alternate authorization identity that did not have the lockdown-mode privilege. The server now requires that both the authentication and authorization identities have the lockdown-mode privilege.

Critical fixes

Critical Fixes

Critical Fixes

This release of the Directory Server addresses critical issues from earlier versions. Update all affected servers appropriately.

- Fixed an issue that could inappropriately allow HTTP-based clients (for example, those using SCIM, the Directory REST API, or the Consent API) to issue requests when the server is in lockdown mode.
  - Fixed in: 8.2.0.0
  - Introduced in: 7.2.0.0
  - Support identifiers: DS-42834
- Fixed an issue where new replicas incorrectly went into lockdown mode after initialization.

  This issue would happen when trying to initialize a newly-added replica to a topology that had been created some time ago. This amount of time had to exceed the replication purge delay, which is 24 hours by default. Before this fix was introduced, you could get past this by running "leave-lockdown-mode" on the new replica, then re-running "dsreplication initialize" on it.
  - Fixed in: 8.2.0.0
  - Introduced in: 8.1.0.0
  - Support identifiers: DS-42790 SF#00695648
- Fixed an issue where mirrored subtree polling could produce config archive files that were identical or ignored the configured insignificant attributes list.
  - Fixed in: 8.1.0.0
  - Introduced in: 7.0.0.0
  - Support identifiers: DS-41762 SF#00675207 SF#00683777
- Addressed an issue that could lead to slow, off-heap memory growth. This only occurred on servers whose cn=Version,cn=monitor entry was retrieved frequently.
  - Fixed in: 8.1.0.0
  - Introduced in: 5.2.0.0
  - Support identifiers: DS-41301
- Addressed an issue where replication could incorrectly detect a backlog that never clears when updating from a pre-7.3 to a 7.3 or later version. This issue requires that servers were previously removed from the topology, and it has been seen rarely.
  - Fixed in: 8.1.0.0
  - Introduced in: 7.3.0.0
  - Support identifiers: DS-40955
- Fixed a memory leak when performing SCIM queries on the Directory Server.
  - Fixed in: 8.1.0.0
  - Introduced in: 7.2.0.0
  - Support identifiers: DS-41206 SF#00681395
- Fixed an issue that could cause the server to report an "Unable to decode a blacklist key" error while trying to open a local DB backend after an unclean shutdown.
  - Fixed in: 8.0.0.0
  - Introduced in: 7.2.0.0
  - Support identifiers: DS-40788

- The following enhancements were made to the topology manager to make it easier to diagnose connection errors:

  - Added monitoring information for all the failed outbound connections (including the time since it's been failing and the last error message seen when the failure occurred) from a server to one of its configured peers and the number of failed outbound connections.

  - Added alarms/alerts for when a server fails to connect to a peer server within a configured grace period.

  - Fixed in: 7.3.0.0
  - Introduced in: 7.0.0.0
  - Support identifiers: DS-38334 SF#00655578
- The topology manager will now raise a mirrored-subtree-manager-connection-asymmetry alarm when a server is able to establish outbound connections to its peer servers, but those peer servers are unable to establish connections back to the server within the configured grace period. The alarm is cleared as soon as there is connection symmetry.

  - Fixed in: 7.3.0.0
  - Introduced in: 7.0.0.0
  - Support identifiers: DS-38344 SF#00655578
- The dsreplication tool has been fixed to work when the node being used to enable replication is currently out-of-sync with the topology master.

  - Fixed in: 7.3.0.0
  - Introduced in: 7.0.0.0
  - Support identifiers: DS-38335 SF#00655578
- Fixed two issues in which the server could have exposed some clear-text passwords in files on the server file system.

  * When creating an encrypted backup of the alarms, alerts, configuration, encryption settings, schema, tasks, or trust store backends, the password used to generate the encryption key (which may have been obtained from an encryption settings definition) could have been inadvertently written into the backup descriptor. This problem does not affect local DB backends (like userRoot), the LDAP changelog backend, or the replication database.

  * When running certain command-line tools with an argument instructing the tool to read a password from a file, the password contained in that file could have been written into the server's tool invocation log instead of the path to that file. Affected tools include backup, create-initial-config, create-initial-proxy-config, dsreplication, enter-lockdown-mode, export-ldif, import-ldif, ldappasswordmodify, leave-lockdown-mode, manage-tasks, manage-topology, migrate-ldap-schema, parallel-update, prepare-endpoint-server, prepare-external-server, realtime-sync, rebuild-index, re-encode-entries, reload-http-connection-handler-certificates, reload-index, remove-defunct-server, restore, rotate-log, and stop-server. Other tools are not affected. Also note that this only includes passwords contained in files that were provided as command-line arguments; passwords included in the tools.properties file, or in a file referenced from tools.properties, would not have been exposed.

  In each of these cases, the files would have been written with permissions that make their contents only accessible to the system account used to run the server. Further, while administrative passwords may have been exposed in the tool invocation log, neither the passwords for regular users, nor any other data from their entries, should have been affected. We have introduced new automated tests to help ensure that such incidents do not occur in the future.

  We recommend changing any administrative passwords you fear may have been compromised as a result of this issue. If you are concerned that the passphrase for an encryption settings definition may have been exposed, then we recommend creating a new encryption settings definition that is preferred for all subsequent encryption operations, exporting your data to LDIF, and re-importing so that it will be encrypted with the new key. You also may wish to re-encrypt or destroy any existing backups, LDIF

exports, or other data encrypted with a compromised key, and you may wish to sanitize or destroy any existing tool invocation log files that may contain clear-text passwords.

  - Fixed in: 7.3.0.0
  - Introduced in: 7.0.0.0
  - Support identifiers: DS-38897 DS-38908
- The following enhancements were made to the topology manager to make it easier to diagnose the connection errors:

  - Added monitoring information for all the failed outbound connections (including the time since it's been failing and the last error message seen when the failure occurred) from a server to one of its configured peers and the number of failed outbound connections.

  - Added alarms/alerts for when a server fails to connect to a peer server within a configured grace period.

  - Fixed in: 7.2.1.0
  - Introduced in: 7.0.0.0
  - Support identifiers: DS-38334 SF#00655578
- The topology manager will now raise a mirrored-subtree-manager-connection-asymmetry alarm when a server is able to establish outbound connections to its peer servers, but those peer servers are unable to establish connections back to the server within the configured grace period. The alarm is cleared when connection symmetry is achieved.

  - Fixed in: 7.2.1.0
  - Introduced in: 7.0.0.0
  - Support identifiers: DS-38344 SF#00655578
- The dsreplication tool has been fixed to work when the node being used to enable replication is currently out-of-sync with the topology master.

  - Fixed in: 7.2.1.0
  - Introduced in: 7.0.0.0
  - Support identifiers: DS-38335 SF#00655578
- Addressed an issue where an InvalidKeyException could occasionally be reported by import-ldif. The error message for this problem resembles, "An unexpected error occurred during merge processing for index 'dc_example_dc_com_sn.equality': InvalidKeyException: The provided passphrase is invalid."

  - Fixed in: 7.2.0.0
  - Introduced in: 7.0.0.0
  - Support identifiers: DS-37313
- Fixed two issues in which the server could have exposed some clear-text passwords in files on the server file system.

  * When creating an encrypted backup of the alarms, alerts, configuration, encryption settings, schema, tasks, or trust store backends, the password used to generate the encryption key (which may have been obtained from an encryption settings definition) could have been inadvertently written into the backup descriptor. This problem does not affect local DB backends (like userRoot), the LDAP changelog backend, or the replication database.

  * When running certain command-line tools with an argument instructing the tool to read a password from a file, the password contained in that file could have been written into the server's tool invocation log instead of the path to that file. Affected tools include backup, create-initial-config, create-initial-proxy-config, dsreplication, enter-lockdown-mode, export-ldif, import-ldif, ldappasswordmodify, leave-lockdown-mode, manage-tasks, manage-topology, migrate-ldap-schema, parallel-update, prepare-endpoint-server, prepare-external-server, realtime-sync, rebuild-index, re-encode-entries, reload-http-connection-handler-certificates, reload-index, remove-defunct-server, restore, rotate-log, and stop-server. Other tools are not affected. Also note that this only includes passwords contained in files that

were provided as command-line arguments; passwords included in the tools.properties file, or in a file referenced from tools.properties, would not have been exposed.

In each of these cases, the files would have been written with permissions that make their contents only accessible to the system account used to run the server. Further, while administrative passwords may have been exposed in the tool invocation log, neither the passwords for regular users, nor any other data from their entries, should have been affected. We have introduced new automated tests to help ensure that such incidents do not occur in the future.

We recommend changing any administrative passwords you fear may have been compromised as a result of this issue. If you are concerned that the passphrase for an encryption settings definition may have been exposed, then we recommend creating a new encryption settings definition that is preferred for all subsequent encryption operations, exporting your data to LDIF, and re-importing so that it will be encrypted with the new key. You also may wish to re-encrypt or destroy any existing backups, LDIF exports, or other data encrypted with a compromised key, and you may wish to sanitize or destroy any existing tool invocation log files that may contain clear-text passwords.

  - Fixed in: 7.0.1.3
  - Introduced in: 7.0.0.0
  - Support identifiers: DS-38897 DS-38908
- Addressed an issue in "dsreplication enable/initialize" that prevented servers from some previous versions (5.2.0.5 and earlier and 6.0.0.*) from initializing newer servers. Servers from these prior versions can now be used to enable replication with current versions of the server.

  - Fixed in: 7.0.0.0
  - Introduced in: 5.2.0.5
  - Support identifiers: DS-35528 SF#624368
- Fixed a very rare race condition with the Frequently Accessed Entry Cache which could lead to an index being marked as degraded and requiring a rebuild.

  The problem is unlikely to happen outside of testing environments since it requires modifying a single entry over 1000 times per second across multiple servers concurrently.

  - Fixed in: 7.0.0.0
  - Introduced in: 5.2.0.6
  - Support identifiers: DS-35616 SF#00625189
- Addressed an issue where an index key could incorrectly be reported as exceeding the index-entry-limit after one billion entries had been imported or added to the directory server. The directory server does not need to contain one billion entries at the same time to be affected by this issue since the entry ID will always increase for each added entry even if entries are deleted. Environments that have experienced this issue should export and reimport their data after applying this patch.

  - Fixed in: 7.0.0.0
  - Introduced in: 2.0.0.0
  - Support identifiers: DS-35790 SF#00625942
- Fixed an issue that could allow users with locked accounts to change their own passwords using the password modify extended operation.

  - Fixed in: 6.2.0.0
  - Introduced in: 5.2.0.3
  - Support identifiers: DS-17074
- Addressed an issue specific to entry-balanced environments where changes received through replication are applied in the incorrect backend. This can occur if a restricted domain is disabled prior to disabling the global domain. With the restricted domain disabled, the affected server could apply the

changes originally targeted for the restricted domain in the global domain. In addition, other servers in the topology will reset their generation ID for the restricted domain.

  - Fixed in: 6.2.0.0
  - Introduced in: 2.1.4.0
  - Support identifiers: DS-17237 SF#3746
- Added an alarm at warning level to notify if any of the important JVM startup arguments are missing or misconfigured.

  - Fixed in: 6.2.0.0
  - Introduced in: 5.0.0.0
  - Support identifiers: DS-12216
- Addressed an issue where a server could incorrectly report missed replication changes at startup in rare circumstances. Server A could report missed changes at startup where

  1) Server B had not received changes directly from a client for a long time (beyond the purge delay),

  2) Since the last successful change, Server B had processed an operation from a client that made it deep enough in the operation processing to generate a change sequence number (CSN) but that operation was later rejected by the server,

  3) Server A is shutdown, and

  4) While Server A is shutdown, the Server B processes one or more changes directly from the client.

  - Fixed in: 6.2.0.0
  - Introduced in: 3.5.0.0
  - Support identifiers: DS-18035 SF#00614612
- Fixed an issue that could prevent the server from properly closing a database transaction under a sustained load of heavily conflicting write operations on a system that is processing those operations at an abnormally slow rate (for example, if the database is not cached and the disk subsystem is completely saturated).

  - Fixed in: 6.2.0.0
  - Introduced in: 6.0.1.0
  - Support identifiers: DS-18070
- Fixed an issue that could allow users with locked accounts to change their own passwords using the password modify extended operation.

  - Fixed in: 6.2.0.0
  - Introduced in: 5.2.0.3
  - Support identifiers: DS-17074
- Addressed an issue specific to entry-balanced environments where changes received through replication are applied in the incorrect backend. This can occur if a restricted domain is disabled prior to disabling the global domain. With the restricted domain disabled, the affected server could apply the changes originally targeted for the restricted domain in the global domain. In addition, other servers in the topology will reset their generation ID for the restricted domain.

  - Fixed in: 6.2.0.0
  - Introduced in: 2.1.4.0
  - Support identifiers: DS-17237 SF#3746
- Added an alarm at warning level to notify if any of the important JVM startup arguments are missing or misconfigured.

  - Fixed in: 6.2.0.0
  - Introduced in: 5.0.0.0
  - Support identifiers: DS-12216

- Addressed an issue where a server could incorrectly report missed replication changes at startup in rare circumstances. Server A could report missed changes at startup where

  1) Server B had not received changes directly from a client for a long time (beyond the purge delay),

  2) Since the last successful change, Server B had processed an operation from a client that made it deep enough in the operation processing to generate a change sequence number (CSN) but that operation was later rejected by the server,

  3) Server A is shutdown, and

  4) While Server A is shutdown, the Server B processes one or more changes directly from the client.

  - Fixed in: 6.2.0.0
  - Introduced in: 3.5.0.0
  - Support identifiers: DS-18035 SF#00614612
- Fixed an issue that could prevent the server from properly closing a database transaction under a sustained load of heavily conflicting write operations on a system that is processing those operations at an abnormally slow rate (for example, if the database is not cached and the disk subsystem is completely saturated).
  - Fixed in: 6.2.0.0
  - Introduced in: 6.0.1.0
  - Support identifiers: DS-18070
- Fixed an issue where opening the backend database might fail with an IllegalStateException that references "exploded-index-background-deletes" when there are several backend exploded indexes.
  - Fixed in: 6.0.0.0
  - Introduced in: 4.6.0.0
  - Support identifiers: DS-15094
- The server can now detect an "out of file handles" situation on the operating system, and shut down to prevent running in an unreliable state.
  - Fixed in: 5.1.0.0
  - Introduced in: 2.1.0.0
  - Support identifiers: DS-12579 SF#2655
- Added a fail safe to the pending changes queue for the Changelog Backend that can detect and ignore recovered changes that do not need to be committed in order to prevent holding up other changes in the queue.
  - Fixed in: 5.0.0.0
  - Introduced in: 4.5.1.0
  - Support identifiers: DS-11720 SF#2453
- Disabled support for SSLv3 by default in the LDAP, HTTP, and JMX connection handlers, and for replication communication. The recently-discovered POODLE vulnerability could potentially allow a network attacker to determine the plaintext behind an SSLv3-encrypted session, which would effectively negate the primary benefit of the encryption.

  SSLv3 was initially defined in 1996, but was supplanted by the release of the TLSv1 definition in 1999 (and subsequently by TLSv1.1 in 2006 and TLSv1.2 in 2008). These newer TLS protocols are not susceptible to the POODLE vulnerability, and the server has supported them (and preferred them over SSLv3) for many years. The act of disabling SSLv3 by default should not have any adverse effect on clients that support any of the newer TLS protocols. However, if there are any legacy client applications that attempt to communicate securely but do not support the newer TLS protocols, they should be updated to support the newer protocols. In the event that there are known clients that do not support any security protocol newer than SSLv3 and that cannot be immediately updated to support a newer protocol, SSLv3 support can be re-enabled using the newly-introduced allowed-insecure-tls-protocol global configuration property. However, since communication using SSLv3 can no longer be considered

secure, it is strongly recommended that every effort be made to update all known clients still using SSLv3.

It is possible to use the server access log to identify LDAP clients that use SSLv3 to communicate with the server. Whenever an LDAP client establishes a secure connection to the server, or whenever a client uses the StartTLS extended operation to secure an existing plaintext connection, the server will generate a SECURITY-NEGOTIATION access log message. The "protocol" element of a SECURITY-NEGOTIATION access log message specifies the name of the security protocol that has been negotiated between the client and the server, and any SECURITY-NEGOTIATION messages with a protocol of "SSLv3" suggest that the associated client is vulnerable to the POODLE attack. In addition, if any connections are terminated for attempting to use the disallowed SSLv3 protocol, the access log message for that disconnect should include a message stating the reason for the termination.

  - Fixed in: 5.0.0.0
  - Introduced in: 2.1.0.0
  - Support identifiers: DS-11782
- Fixed a problem that could interfere with access to an exploded attribute index after performing an online index rebuild for that attribute.
  - Fixed in: 4.6.0.0
  - Introduced in: 4.5.1.0
  - Support identifiers: DS-10470
- Fix a bug in low level protocol buffer that could result in "uncaught exception" errors.
  - Fixed in: 4.5.0.0
  - Introduced in: 3.2.0.0
  - Support identifiers: DS-9268 SF#2002
- Improve server stability by disabling explicit garbage collections that were being caused by JMX connections.
  - Fixed in: 4.0.0.0
  - Introduced in: 3.5.0.0
  - Support identifiers: DS-7633
- Fix a bug in the LDAP Changelog where the changelog index manager could capture new changes for an attribute in one index after already hitting the end of another index. This created the possibility for changes to be missed when processing get-changelog-batch-requests at the same time that live traffic is happening.
  - Fixed in: 3.6.0.0
  - Introduced in: 3.2.0.0
  - Support identifiers: DS-7422
- Fix a bug that allows users with expired passwords to change attributes in their own entry other than password.
  - Fixed in: 3.5.0.0
  - Introduced in: 3.2.0.0
  - Support identifiers: DS-6054
- Address an issue where a directory server might resend duplicate changes when processing a GetChangelogBatch request in an environment that is under heavy load.
  - Fixed in: 3.5.0.0
  - Introduced in: 3.2.0.0
  - Support identifiers: DS-5656

- Update the PingDirectory Server to apply access controls when processing the GetAuthorizationEntryRequestControl.
  - Fixed in: 3.5.0.0
  - Introduced in: 2.0.0.0
  - Support identifiers: DS-854
- Fix a bug where PingDirectory Servers could potentially miss some update messages in large topologies after a restart.
  - Fixed in: 3.2.0.0
  - Introduced in: 3.1.0.0
  - Support identifiers: DS-3592

This release of PingDirectory Server addresses critical issues from earlier versions. Update all affected servers appropriately.

- Fixed an issue where new replicas incorrectly went into lockdown mode after initialization.

  This issue would happen when trying to initialize a newly-added replica to a topology that had been created some time ago. This amount of time had to exceed the replication purge delay, which is 24 hours by default. Before this fix was introduced, you could get past this by running "leave-lockdown-mode" on the new replica, then re-running "dsreplication initialize" on it.

  Fixed in: 8.2.0.0

  Introduced in: 8.1.0.0

  Support identifiers: DS-42790 SF#00695648
- Addressed an issue that could lead to slow, off-heap memory growth. This only occurred on servers whose cn=Version,cn=monitor entry was retrieved frequently.
  - Fixed in: 8.1.0.0
  - Introduced in: 5.2.0.0
  - Support identifiers: DS-41301
- Addressed an issue where replication could incorrectly detect a backlog that never clears when updating from a pre-7.3 to a 7.3 or later version. This issue requires that servers were previously removed from the topology, and it has been seen rarely.
  - Fixed in: 8.1.0.0
  - Introduced in: 7.3.0.0
  - Support identifiers: DS-40955
- Fixed a memory leak when performing SCIM queries on the PingDirectory Server.
  - Fixed in: 8.1.0.0
  - Introduced in: 7.2.0.0
  - Support identifiers: DS-41206 SF#00681395
- Fixed an issue that could cause the server to report an "Unable to decode a blacklist key" error while trying to open a local DB backend after an unclean shutdown.
  - Fixed in: 8.0.0.0
  - Introduced in: 7.2.0.0
  - Support identifiers: DS-40788

- The following enhancements were made to the topology manager to make it easier to diagnose connection errors:

  - Added monitoring information for all the failed outbound connections (including the time since it's been failing and the last error message seen when the failure occurred) from a server to one of its configured peers and the number of failed outbound connections.

  - Added alarms/alerts for when a server fails to connect to a peer server within a configured grace period.

  - Fixed in: 7.3.0.0
  - Introduced in: 7.0.0.0
  - Support identifiers: DS-38334 SF#00655578
- The topology manager will now raise a mirrored-subtree-manager-connection-asymmetry alarm when a server is able to establish outbound connections to its peer servers, but those peer servers are unable to establish connections back to the server within the configured grace period. The alarm is cleared as soon as there is connection symmetry.

  - Fixed in: 7.3.0.0
  - Introduced in: 7.0.0.0
  - Support identifiers: DS-38344 SF#00655578
- The dsreplication tool has been fixed to work when the node being used to enable replication is currently out-of-sync with the topology master.

  - Fixed in: 7.3.0.0
  - Introduced in: 7.0.0.0
  - Support identifiers: DS-38335 SF#00655578
- Fixed two issues in which the server could have exposed some clear-text passwords in files on the server file system.

  * When creating an encrypted backup of the alarms, alerts, configuration, encryption settings, schema, tasks, or trust store backends, the password used to generate the encryption key (which may have been obtained from an encryption settings definition) could have been inadvertently written into the backup descriptor. This problem does not affect local DB backends (like userRoot), the LDAP changelog backend, or the replication database.

  * When running certain command-line tools with an argument instructing the tool to read a password from a file, the password contained in that file could have been written into the server's tool invocation log instead of the path to that file. Affected tools include backup, create-initial-config, create-initial-proxy-config, dsreplication, enter-lockdown-mode, export-ldif, import-ldif, ldappasswordmodify, leave-lockdown-mode, manage-tasks, manage-topology, migrate-ldap-schema, parallel-update, prepare-endpoint-server, prepare-external-server, realtime-sync, rebuild-index, re-encode-entries, reload-http-connection-handler-certificates, reload-index, remove-defunct-server, restore, rotate-log, and stop-server. Other tools are not affected. Also note that this only includes passwords contained in files that were provided as command-line arguments; passwords included in the tools.properties file, or in a file referenced from tools.properties, would not have been exposed.

  In each of these cases, the files would have been written with permissions that make their contents only accessible to the system account used to run the server. Further, while administrative passwords may have been exposed in the tool invocation log, neither the passwords for regular users, nor any other data from their entries, should have been affected. We have introduced new automated tests to help ensure that such incidents do not occur in the future.

  We recommend changing any administrative passwords you fear may have been compromised as a result of this issue. If you are concerned that the passphrase for an encryption settings definition may have been exposed, then we recommend creating a new encryption settings definition that is preferred for all subsequent encryption operations, exporting your data to LDIF, and re-importing so that it will be encrypted with the new key. You also may wish to re-encrypt or destroy any existing backups, LDIF

exports, or other data encrypted with a compromised key, and you may wish to sanitize or destroy any existing tool invocation log files that may contain clear-text passwords.

  - Fixed in: 7.3.0.0
  - Introduced in: 7.0.0.0
  - Support identifiers: DS-38897 DS-38908
- The following enhancements were made to the topology manager to make it easier to diagnose the connection errors:

  - Added monitoring information for all the failed outbound connections (including the time since it's been failing and the last error message seen when the failure occurred) from a server to one of its configured peers and the number of failed outbound connections.

  - Added alarms/alerts for when a server fails to connect to a peer server within a configured grace period.

  - Fixed in: 7.2.1.0
  - Introduced in: 7.0.0.0
  - Support identifiers: DS-38334 SF#00655578
- The topology manager will now raise a mirrored-subtree-manager-connection-asymmetry alarm when a server is able to establish outbound connections to its peer servers, but those peer servers are unable to establish connections back to the server within the configured grace period. The alarm is cleared when connection symmetry is achieved.

  - Fixed in: 7.2.1.0
  - Introduced in: 7.0.0.0
  - Support identifiers: DS-38344 SF#00655578
- The dsreplication tool has been fixed to work when the node being used to enable replication is currently out-of-sync with the topology master.

  - Fixed in: 7.2.1.0
  - Introduced in: 7.0.0.0
  - Support identifiers: DS-38335 SF#00655578
- Addressed an issue where an InvalidKeyException could occasionally be reported by import-ldif. The error message for this problem resembles, "An unexpected error occurred during merge processing for index 'dc_example_dc_com_sn.equality': InvalidKeyException: The provided passphrase is invalid."

  - Fixed in: 7.2.0.0
  - Introduced in: 7.0.0.0
  - Support identifiers: DS-37313
- Fixed two issues in which the server could have exposed some clear-text passwords in files on the server file system.

  * When creating an encrypted backup of the alarms, alerts, configuration, encryption settings, schema, tasks, or trust store backends, the password used to generate the encryption key (which may have been obtained from an encryption settings definition) could have been inadvertently written into the backup descriptor. This problem does not affect local DB backends (like userRoot), the LDAP changelog backend, or the replication database.

  * When running certain command-line tools with an argument instructing the tool to read a password from a file, the password contained in that file could have been written into the server's tool invocation log instead of the path to that file. Affected tools include backup, create-initial-config, create-initial-proxy-config, dsreplication, enter-lockdown-mode, export-ldif, import-ldif, ldappasswordmodify, leave-lockdown-mode, manage-tasks, manage-topology, migrate-ldap-schema, parallel-update, prepare-endpoint-server, prepare-external-server, realtime-sync, rebuild-index, re-encode-entries, reload-http-connection-handler-certificates, reload-index, remove-defunct-server, restore, rotate-log, and stop-server. Other tools are not affected. Also note that this only includes passwords contained in files that

were provided as command-line arguments; passwords included in the tools.properties file, or in a file referenced from tools.properties, would not have been exposed.

In each of these cases, the files would have been written with permissions that make their contents only accessible to the system account used to run the server. Further, while administrative passwords may have been exposed in the tool invocation log, neither the passwords for regular users, nor any other data from their entries, should have been affected. We have introduced new automated tests to help ensure that such incidents do not occur in the future.

We recommend changing any administrative passwords you fear may have been compromised as a result of this issue. If you are concerned that the passphrase for an encryption settings definition may have been exposed, then we recommend creating a new encryption settings definition that is preferred for all subsequent encryption operations, exporting your data to LDIF, and re-importing so that it will be encrypted with the new key. You also may wish to re-encrypt or destroy any existing backups, LDIF exports, or other data encrypted with a compromised key, and you may wish to sanitize or destroy any existing tool invocation log files that may contain clear-text passwords.

  - Fixed in: 7.0.1.3
  - Introduced in: 7.0.0.0
  - Support identifiers: DS-38897 DS-38908
- Addressed an issue in "dsreplication enable/initialize" that prevented servers from some previous versions (5.2.0.5 and earlier and 6.0.0.*) from initializing newer servers. Servers from these prior versions can now be used to enable replication with current versions of the server.

  - Fixed in: 7.0.0.0
  - Introduced in: 5.2.0.5
  - Support identifiers: DS-35528 SF#624368
- Fixed a very rare race condition with the Frequently Accessed Entry Cache which could lead to an index being marked as degraded and requiring a rebuild.

  The problem is unlikely to happen outside of testing environments since it requires modifying a single entry over 1000 times per second across multiple servers concurrently.

  - Fixed in: 7.0.0.0
  - Introduced in: 5.2.0.6
  - Support identifiers: DS-35616 SF#00625189
- Addressed an issue where an index key could incorrectly be reported as exceeding the index-entry-limit after one billion entries had been imported or added to the directory server. The directory server does not need to contain one billion entries at the same time to be affected by this issue since the entry ID will always increase for each added entry even if entries are deleted. Environments that have experienced this issue should export and reimport their data after applying this patch.

  - Fixed in: 7.0.0.0
  - Introduced in: 2.0.0.0
  - Support identifiers: DS-35790 SF#00625942
- Fixed an issue that could allow users with locked accounts to change their own passwords using the password modify extended operation.

  - Fixed in: 6.2.0.0
  - Introduced in: 5.2.0.3
  - Support identifiers: DS-17074
- Addressed an issue specific to entry-balanced environments where changes received through replication are applied in the incorrect backend. This can occur if a restricted domain is disabled prior to disabling the global domain. With the restricted domain disabled, the affected server could apply the

changes originally targeted for the restricted domain in the global domain. In addition, other servers in the topology will reset their generation ID for the restricted domain.

  - Fixed in: 6.2.0.0
  - Introduced in: 2.1.4.0
  - Support identifiers: DS-17237 SF#3746
- Added an alarm at warning level to notify if any of the important JVM startup arguments are missing or misconfigured.

  - Fixed in: 6.2.0.0
  - Introduced in: 5.0.0.0
  - Support identifiers: DS-12216
- Addressed an issue where a server could incorrectly report missed replication changes at startup in rare circumstances. Server A could report missed changes at startup where

  1) Server B had not received changes directly from a client for a long time (beyond the purge delay),

  2) Since the last successful change, Server B had processed an operation from a client that made it deep enough in the operation processing to generate a change sequence number (CSN) but that operation was later rejected by the server,

  3) Server A is shutdown, and

  4) While Server A is shutdown, the Server B processes one or more changes directly from the client.

  - Fixed in: 6.2.0.0
  - Introduced in: 3.5.0.0
  - Support identifiers: DS-18035 SF#00614612
- Fixed an issue that could prevent the server from properly closing a database transaction under a sustained load of heavily conflicting write operations on a system that is processing those operations at an abnormally slow rate (for example, if the database is not cached and the disk subsystem is completely saturated).

  - Fixed in: 6.2.0.0
  - Introduced in: 6.0.1.0
  - Support identifiers: DS-18070
- Fixed an issue that could allow users with locked accounts to change their own passwords using the password modify extended operation.

  - Fixed in: 6.2.0.0
  - Introduced in: 5.2.0.3
  - Support identifiers: DS-17074
- Addressed an issue specific to entry-balanced environments where changes received through replication are applied in the incorrect backend. This can occur if a restricted domain is disabled prior to disabling the global domain. With the restricted domain disabled, the affected server could apply the changes originally targeted for the restricted domain in the global domain. In addition, other servers in the topology will reset their generation ID for the restricted domain.

  - Fixed in: 6.2.0.0
  - Introduced in: 2.1.4.0
  - Support identifiers: DS-17237 SF#3746
- Added an alarm at warning level to notify if any of the important JVM startup arguments are missing or misconfigured.

  - Fixed in: 6.2.0.0
  - Introduced in: 5.0.0.0
  - Support identifiers: DS-12216

- Addressed an issue where a server could incorrectly report missed replication changes at startup in rare circumstances. Server A could report missed changes at startup where

  1) Server B had not received changes directly from a client for a long time (beyond the purge delay),

  2) Since the last successful change, Server B had processed an operation from a client that made it deep enough in the operation processing to generate a change sequence number (CSN) but that operation was later rejected by the server,

  3) Server A is shutdown, and

  4) While Server A is shutdown, the Server B processes one or more changes directly from the client.

  - Fixed in: 6.2.0.0
  - Introduced in: 3.5.0.0
  - Support identifiers: DS-18035 SF#00614612
- Fixed an issue that could prevent the server from properly closing a database transaction under a sustained load of heavily conflicting write operations on a system that is processing those operations at an abnormally slow rate (for example, if the database is not cached and the disk subsystem is completely saturated).
  - Fixed in: 6.2.0.0
  - Introduced in: 6.0.1.0
  - Support identifiers: DS-18070
- Fixed an issue where opening the backend database might fail with an IllegalStateException that references "exploded-index-background-deletes" when there are several backend exploded indexes.
  - Fixed in: 6.0.0.0
  - Introduced in: 4.6.0.0
  - Support identifiers: DS-15094
- The server can now detect an "out of file handles" situation on the operating system, and shut down to prevent running in an unreliable state.
  - Fixed in: 5.1.0.0
  - Introduced in: 2.1.0.0
  - Support identifiers: DS-12579 SF#2655
- Added a fail safe to the pending changes queue for the Changelog Backend that can detect and ignore recovered changes that do not need to be committed in order to prevent holding up other changes in the queue.
  - Fixed in: 5.0.0.0
  - Introduced in: 4.5.1.0
  - Support identifiers: DS-11720 SF#2453
- Disabled support for SSLv3 by default in the LDAP, HTTP, and JMX connection handlers, and for replication communication. The recently-discovered POODLE vulnerability could potentially allow a network attacker to determine the plaintext behind an SSLv3-encrypted session, which would effectively negate the primary benefit of the encryption.

  SSLv3 was initially defined in 1996, but was supplanted by the release of the TLSv1 definition in 1999 (and subsequently by TLSv1.1 in 2006 and TLSv1.2 in 2008). These newer TLS protocols are not susceptible to the POODLE vulnerability, and the server has supported them (and preferred them over SSLv3) for many years. The act of disabling SSLv3 by default should not have any adverse effect on clients that support any of the newer TLS protocols. However, if there are any legacy client applications that attempt to communicate securely but do not support the newer TLS protocols, they should be updated to support the newer protocols. In the event that there are known clients that do not support any security protocol newer than SSLv3 and that cannot be immediately updated to support a newer protocol, SSLv3 support can be re-enabled using the newly-introduced allowed-insecure-tls-protocol global configuration property. However, since communication using SSLv3 can no longer be considered

secure, it is strongly recommended that every effort be made to update all known clients still using SSLv3.

It is possible to use the server access log to identify LDAP clients that use SSLv3 to communicate with the server. Whenever an LDAP client establishes a secure connection to the server, or whenever a client uses the StartTLS extended operation to secure an existing plaintext connection, the server will generate a SECURITY-NEGOTIATION access log message. The "protocol" element of a SECURITY-NEGOTIATION access log message specifies the name of the security protocol that has been negotiated between the client and the server, and any SECURITY-NEGOTIATION messages with a protocol of "SSLv3" suggest that the associated client is vulnerable to the POODLE attack. In addition, if any connections are terminated for attempting to use the disallowed SSLv3 protocol, the access log message for that disconnect should include a message stating the reason for the termination.

  - Fixed in: 5.0.0.0
  - Introduced in: 2.1.0.0
  - Support identifiers: DS-11782
- Fixed a problem that could interfere with access to an exploded attribute index after performing an online index rebuild for that attribute.
  - Fixed in: 4.6.0.0
  - Introduced in: 4.5.1.0
  - Support identifiers: DS-10470
- Fix a bug in low level protocol buffer that could result in "uncaught exception" errors.
  - Fixed in: 4.5.0.0
  - Introduced in: 3.2.0.0
  - Support identifiers: DS-9268 SF#2002
- Improve server stability by disabling explicit garbage collections that were being caused by JMX connections.
  - Fixed in: 4.0.0.0
  - Introduced in: 3.5.0.0
  - Support identifiers: DS-7633
- Fix a bug in the LDAP Changelog where the changelog index manager could capture new changes for an attribute in one index after already hitting the end of another index. This created the possibility for changes to be missed when processing get-changelog-batch-requests at the same time that live traffic is happening.
  - Fixed in: 3.6.0.0
  - Introduced in: 3.2.0.0
  - Support identifiers: DS-7422
- Fix a bug that allows users with expired passwords to change attributes in their own entry other than password.
  - Fixed in: 3.5.0.0
  - Introduced in: 3.2.0.0
  - Support identifiers: DS-6054
- Address an issue where a directory server might resend duplicate changes when processing a GetChangelogBatch request in an environment that is under heavy load.
  - Fixed in: 3.5.0.0
  - Introduced in: 3.2.0.0
  - Support identifiers: DS-5656

- Update the PingDirectory Server to apply access controls when processing the GetAuthorizationEntryRequestControl.
  - Fixed in: 3.5.0.0
  - Introduced in: 2.0.0.0
  - Support identifiers: DS-854
- Fix a bug where PingDirectory Servers could potentially miss some update messages in large topologies after a restart.
  - Fixed in: 3.2.0.0
  - Introduced in: 3.1.0.0
  - Support identifiers: DS-3592

Release Notes Archive

Release notes for previous versions of PingDirectory Server are included for reference.

PingDirectory Server 8.1.0.6 release notes
The release notes for the 8.1.0.6 release of PingDirectory Server.

Critical fixes

This release of the Directory Server addresses critical issues from earlier versions. Update all affected servers appropriately.

No critical issues have been identified.

Resolved issues

The following issues have been resolved with this release of the Directory Server.

Ticket ID	Description
DS-42961	To help with replication backlog analysis, the replication summary monitor now includes a 'replica-partial-backlog' attribute that shows how each origin replica contributes partial backlog with the 'per-origin-replication-backlog' property. The 'replica-partial-backlog' attribute also shows the change numbers used for the calculation.
DS-43898	Addressed an issue where the backend would store an incorrect count for a database key that had exceeded the index entry limit. This only happened for composite indexes when the server started up after not being shut down cleanly. Normal search processing was unaffected, but the matching entry count request control processing would return incorrect results.

Ticket ID	Description
DS-43959, DS-44924	Added new global configuration properties that can be used to impose limits on the maximum number of attributes that can be present in an **`add`** request and the maximum number of modifications in a **`modify`** request, which can be used to avoid potential denial of service attacks. Both limits are set to 1000 by default, which is likely to be adequate for all legitimate use cases, and neither property affects the number of values that can be provided for an attribute.

PingDirectory Server 8.1.0.5 release notes

Critical fixes

This release of the Directory Server addresses critical issues from earlier versions. Update all affected servers appropriately.

No critical issues have been identified.

Resolved issues

The following issues have been resolved with this release of the Directory Server.

Ticket ID	Description
DS-43288	Updated setup and the replace-certificate tool to improve the way we generate self-signed certificates and certificate signing requests to make them more palatable to clients. To reduce the frequency with which administrators had to replace self-signed certificates, we previously used a very long lifetime for self-signed certificates generated by setup or the replace-certificate tool. However, some clients (especially web browsers and other HTTP clients) have started more strenuously objecting to certificates to long lifetimes, so we now generate self-signed certificates with a one-year validity period. The inter-server certificate (which is used internally within the server and does not get exposed to normal clients) is still created with a twenty-year lifetime. Also, the replace-certificate tool's interactive mode has been updated to improve the process that it uses to obtain information to include in the subject DN and subject alternative name extension for self-signed certificates and certificate signing requests. The following changes have been made in accordance with CA/Browser Forum guidelines: * When selecting the subject DN for the certificate, we listed a number of common attributes that may be used, including CN, OU, O, L, ST, and C. We previously indicated that CN attribute was recommended. We now also indicate that the O and C attributes are recommended as well. * When obtaining the list of DNS names to include in the subject alternative name extension, we previously suggested all names that we could find associated with interfaces on the local system. In many cases, we now omit non-qualified names and names that are associated with loopback interfaces. We will also warn about any attempts to add unqualified or invalid names to the list. * When obtaining the list of IP addresses to include in the subject alternative name extension, we previously suggested all addresses associated with all network interfaces on the system. We no longer suggest any IP addresses associated with loopback interfaces, and we no longer suggest any IP addresses associated in IANA-reserved ranges (for example, addresses reserved for private-use networks). The tool will now warn about attempts to add these addresses for inclusion in the subject alternative name extension.

Ticket ID	Description
DS-43576	Added the ability to retry operation failures from replication if the failures are likely due to dependent writes being played out of order. This issue only affected environments that were sending writes to different servers, and also were not able to use the appropriate level of replication assurance. To enable this setting, update the on-replay-failure-wait-for-dependent-ops-timeout configuration property on a replication domain.
DS-44037	Avoid lockdown due to missing changes during enable caused by a missing timestamp that indicates the enable time. The problem resulted in change numbers and error messages with dates around the year 1970.
DS-44204	Fixed an issue in which the Directory Server could incorrectly allow requests to be processed with an alternate authorization identity (for example, using the proxied authorization control, or if the requests pass through a Directory Proxy Server) whose account is in a "must change password" state. The server will now only permit the operation if it attempts to set a new password for the target user.
DS-44316	Reduced the JVM memory requirements for many command line tools. This avoids memory pressure when multiple tools, such as a scheduled collect-support-data task, are run concurrently to the server process. For most tools, the initial heap size has been reduced to 128 MB, and for certain tools the maximum heap size has capped at 512 MB. On systems with larger amounts of memory, these tools previously were allotted unnecessarily large heaps. The maximum heap size has not been reduced for any tool that especially benefits from having more memory.

PingDirectory Server 8.1.0.2 Release Notes

Critical Fixes

This release of PingDirectory Server addresses critical issues from earlier versions. Update all affected servers appropriately.

- Fixed an issue where new replicas incorrectly went into lockdown mode after initialization. This issue would happen when trying to initialize a newly-added replica to a topology that had been created some time ago. This amount of time had to exceed the replication purge delay, which is 24 hours by default. Before this fix was introduced, you could get past this by running "leave-lockdown-mode" on the new replica, then re-running "dsreplication initialize" on it.
  - Fixed in: 8.1.0.2
  - Introduced in: 8.1.0.0
  - Support identifiers: DS-42790 SF#00695648

- Addressed an issue that could lead to slow, off-heap memory growth. This only occurred on servers whose cn=Version,cn=monitor entry was retrieved frequently.
  - Fixed in: 8.1.0.0
  - Introduced in: 5.2.0.0
  - Support identifiers: DS-41301
- Addressed an issue where replication could incorrectly detect a backlog that never clears when updating from a pre-7.3 to a 7.3 or later version. This issue requires that servers were previously removed from the topology, and it has been seen rarely.
  - Fixed in: 8.1.0.0
  - Introduced in: 7.3.0.0
  - Support identifiers: DS-40955
- Fixed a memory leak when performing SCIM queries on PingDirectory Server.
  - Fixed in: 8.1.0.0
  - Introduced in: 7.2.0.0
  - Support identifiers: DS-41206 SF#00681395
- Fixed an issue that could cause the server to report an "Unable to decode a blacklist key" error while trying to open a local DB backend after an unclean shutdown.
  - Fixed in: 8.0.0.0
  - Introduced in: 7.2.0.0
  - Support identifiers: DS-40788
- The following enhancements were made to the topology manager to make it easier to diagnose connection errors:
  - Added monitoring information for all the failed outbound connections (including the time since it has been failing and the last error message seen when the failure occurred) from a server to one of its configured peers and the number of failed outbound connections.
  - Added alarms/alerts for when a server fails to connect to a peer server within a configured grace period.
  - Fixed in: 7.3.0.0
  - Introduced in: 7.0.0.0
  - Support identifiers: DS-38334 SF#00655578
- The topology manager will now raise a mirrored-subtree-manager-connection-asymmetry alarm when a server is able to establish outbound connections to its peer servers, but those peer servers are unable to establish connections back to the server within the configured grace period. The alarm is cleared as soon as there is connection symmetry.
  - Fixed in: 7.3.0.0
  - Introduced in: 7.0.0.0
  - Support identifiers: DS-38344 SF#00655578
- The dsreplication tool has been fixed to work when the node being used to enable replication is currently out-of-sync with the topology master.
  - Fixed in: 7.3.0.0
  - Introduced in: 7.0.0.0
  - Support identifiers: DS-38335 SF#00655578
- Fixed two issues in which the server could have exposed some clear-text passwords in files on the server file system.
  - Fixed in: 7.3.0.0
  - Introduced in: 7.0.0.0
  - Support identifiers: DS-38897 DS-38908

- The following enhancements were made to the topology manager to make it easier to diagnose the connection errors:
  - Fixed in: 7.2.1.0
  - Introduced in: 7.0.0.0
  - Support identifiers: DS-38334 SF#00655578
- The topology manager will now raise a mirrored-subtree-manager-connection-asymmetry alarm when a server is able to establish outbound connections to its peer servers, but those peer servers are unable to establish connections back to the server within the configured grace period. The alarm is cleared when connection symmetry is achieved.
  - Fixed in: 7.2.1.0
  - Introduced in: 7.0.0.0
  - Support identifiers: DS-38344 SF#00655578
- The dsreplication tool has been fixed to work when the node being used to enable replication is currently out-of-sync with the topology master.
  - Fixed in: 7.2.1.0
  - Introduced in: 7.0.0.0
  - Support identifiers: DS-38335 SF#00655578
- Addressed an issue where an InvalidKeyException could occasionally be reported by import-ldif. The error message for this problem resembles, "An unexpected error occurred during merge processing for index 'dc_example_dc_com_sn.equality': InvalidKeyException: The provided passphrase is invalid."
  - Fixed in: 7.2.0.0
  - Introduced in: 7.0.0.0
  - Support identifiers: DS-37313
- Fixed two issues in which the server could have exposed some clear-text passwords in files on the server file system.
  - Fixed in: 7.0.1.3
  - Introduced in: 7.0.0.0
  - Support identifiers: DS-38897 DS-38908
- Addressed an issue in "dsreplication enable/initialize" that prevented servers from some previous versions (5.2.0.5 and earlier and 6.0.0.*) from initializing newer servers. Servers from these prior versions can now be used to enable replication with current versions of the server.
  - Fixed in: 7.0.0.0
  - Introduced in: 5.2.0.5
  - Support identifiers: DS-35528 SF#624368
- Fixed a very rare race condition with the Frequently Accessed Entry Cache which could lead to an index being marked as degraded and requiring a rebuild.
  - Fixed in: 7.0.0.0
  - Introduced in: 5.2.0.6
  - Support identifiers: DS-35616 SF#00625189
- Addressed an issue where an index key could incorrectly be reported as exceeding the index-entry-limit after one billion entries had been imported or added to the directory server. The directory server does not need to contain one billion entries at the same time to be affected by this issue since the entry ID will always increase for each added entry even if entries are deleted. Environments that have experienced this issue should export and reimport their data after applying this patch.
  - Fixed in: 7.0.0.0
  - Introduced in: 2.0.0.0
  - Support identifiers: DS-35790 SF#00625942

- Fixed an issue that could allow users with locked accounts to change their own passwords using the password modify extended operation.
  - Fixed in: 6.2.0.0
  - Introduced in: 5.2.0.3
  - Support identifiers: DS-17074
- Addressed an issue specific to entry-balanced environments where changes received through replication are applied in the incorrect backend. This can occur if a restricted domain is disabled prior to disabling the global domain. With the restricted domain disabled, the affected server could apply the changes originally targeted for the restricted domain in the global domain. In addition, other servers in the topology will reset their generation ID for the restricted domain.
  - Fixed in: 6.2.0.0
  - Introduced in: 2.1.4.0
  - Support identifiers: DS-17237 SF#3746
- Added an alarm at warning level to notify if any of the important JVM startup arguments are missing or misconfigured.
  - Fixed in: 6.2.0.0
  - Introduced in: 5.0.0.0
  - Support identifiers: DS-12216
- Addressed an issue where a server could incorrectly report missed replication changes at startup in rare circumstances. Server A could report missed changes at startup where
  - Fixed in: 6.2.0.0
  - Introduced in: 3.5.0.0
  - Support identifiers: DS-18035 SF#00614612
- Fixed an issue that could prevent the server from properly closing a database transaction under a sustained load of heavily conflicting write operations on a system that is processing those operations at an abnormally slow rate (for example, if the database is not cached and the disk subsystem is completely saturated).
  - Fixed in: 6.2.0.0
  - Introduced in: 6.0.1.0
  - Support identifiers: DS-18070
- Fixed an issue that could allow users with locked accounts to change their own passwords using the password modify extended operation.
  - Fixed in: 6.2.0.0
  - Introduced in: 5.2.0.3
  - Support identifiers: DS-17074
- Addressed an issue specific to entry-balanced environments where changes received through replication are applied in the incorrect backend. This can occur if a restricted domain is disabled prior to disabling the global domain. With the restricted domain disabled, the affected server could apply the changes originally targeted for the restricted domain in the global domain. In addition, other servers in the topology will reset their generation ID for the restricted domain.
  - Fixed in: 6.2.0.0
  - Introduced in: 2.1.4.0
  - Support identifiers: DS-17237 SF#3746
- Added an alarm at warning level to notify if any of the important JVM startup arguments are missing or misconfigured.
  - Fixed in: 6.2.0.0
  - Introduced in: 5.0.0.0
  - Support identifiers: DS-12216

- Addressed an issue where a server could incorrectly report missed replication changes at startup in rare circumstances. Server A could report missed changes at startup where
  - Fixed in: 6.2.0.0
  - Introduced in: 3.5.0.0
  - Support identifiers: DS-18035 SF#00614612
- Fixed an issue that could prevent the server from properly closing a database transaction under a sustained load of heavily conflicting write operations on a system that is processing those operations at an abnormally slow rate (for example, if the database is not cached and the disk subsystem is completely saturated).
  - Fixed in: 6.2.0.0
  - Introduced in: 6.0.1.0
  - Support identifiers: DS-18070
- Fixed an issue where opening the backend database might fail with an IllegalStateException that references "exploded-index-background-deletes" when there are several backend exploded indexes.
  - Fixed in: 6.0.0.0
  - Introduced in: 4.6.0.0
  - Support identifiers: DS-15094
- The server can now detect an "out of file handles" situation on the operating system, and shut down to prevent running in an unreliable state.
  - Fixed in: 5.1.0.0
  - Introduced in: 2.1.0.0
  - Support identifiers: DS-12579 SF#2655
- Added a fail safe to the pending changes queue for the Changelog Backend that can detect and ignore recovered changes that do not need to be committed in order to prevent holding up other changes in the queue.
  - Fixed in: 5.0.0.0
  - Introduced in: 4.5.1.0
  - Support identifiers: DS-11720 SF#2453
- Disabled support for SSLv3 by default in the LDAP, HTTP, and JMX connection handlers, and for replication communication. The recently-discovered POODLE vulnerability could potentially allow a network attacker to determine the plaintext behind an SSLv3-encrypted session, which would effectively negate the primary benefit of the encryption.
  - Fixed in: 5.0.0.0
  - Introduced in: 2.1.0.0
  - Support identifiers: DS-11782
- Fixed a problem that could interfere with access to an exploded attribute index after performing an online index rebuild for that attribute.
  - Fixed in: 4.6.0.0
  - Introduced in: 4.5.1.0
  - Support identifiers: DS-10470
- Fix a bug in low level protocol buffer that could result in "uncaught exception" errors.
  - Fixed in: 4.5.0.0
  - Introduced in: 3.2.0.0
  - Support identifiers: DS-9268 SF#2002

- Improve server stability by disabling explicit garbage collections that were being caused by JMX connections.
  - Fixed in: 4.0.0.0
  - Introduced in: 3.5.0.0
  - Support identifiers: DS-7633
- Fix a bug in the LDAP Changelog where the changelog index manager could capture new changes for an attribute in one index after already hitting the end of another index. This created the possibility for changes to be missed when processing get-changelog-batch-requests at the same time that live traffic is happening.
  - Fixed in: 3.6.0.0
  - Introduced in: 3.2.0.0
  - Support identifiers: DS-7422
- Fix a bug that allows users with expired passwords to change attributes in their own entry other than password.
  - Fixed in: 3.5.0.0
  - Introduced in: 3.2.0.0
  - Support identifiers: DS-6054
- Address an issue where a directory server might resend duplicate changes when processing a GetChangelogBatch request in an environment that is under heavy load.
  - Fixed in: 3.5.0.0
  - Introduced in: 3.2.0.0
  - Support identifiers: DS-5656
- Update PingDirectory Server to apply access controls when processing the GetAuthorizationEntryRequestControl.
  - Fixed in: 3.5.0.0
  - Introduced in: 2.0.0.0
  - Support identifiers: DS-854
- Fix a bug where PingDirectory Servers could potentially miss some update messages in large topologies after a restart.
  - Fixed in: 3.2.0.0
  - Introduced in: 3.1.0.0
  - Support identifiers: DS-3592

Resolved Issues

The following issues have been resolved with this release of PingDirectory Server:

Ticket ID	Description
DS-40828	Fixed an issue where some state associated with a JMX connection was not freed after the connection was closed. This led to a slow memory leak in servers that were monitored by an application that created a new JMX connection each polling interval.
DS-41964	Fixed an issue with the manage-profile tool where files in a server profile's dsconfig/ directory without a ".dsconfig" extension could cause failures in manage-profile replace-profile when validating updated dsconfig files.

Ticket ID	Description
DS-42438	Fixed an issue that could cause the remove-defunct-server tool to not remove certain replication attributes when run with a topology json file.
DS-42687	Upgrade to Jetty 9.4.30
DS-42790	Fixed an issue where new replicas incorrectly went into lockdown mode after initialization. This issue would happen when trying to initialize a newly-added replica to a topology that had been created some time ago. This amount of time had to exceed the replication purge delay, which is 24 hours by default. Before this fix was introduced, you could get past this by running "leave-lockdown-mode" on the new replica, then re-running "dsreplication initialize" on it.

PingDirectory Server 8.1 Release Notes

Upgrade Considerations

Important considerations for upgrading to this version of PingDirectory Server:

> ⓘ **Important:**
>
> If you plan to upgrade servers using a mixed-version environment where one version is earlier than 7.0 and some of the servers are still using the admin backend while others have been updated to the topology registry, do not attempt to make size changes to the topology. You cannot remove any existing servers (using `dsreplication disable`) or add new servers (using `dsreplication enable`) when in this transitional state of partially-updated servers. When a topology has been completely migrated to a 7.0 or later version with the topology registry, changes to the topology size are allowed, even in mixed-version environments (for example, mixed 7.3 and 8.3).

- If you have upgraded a server that is in a cluster (that has a cluster name set in the Server Instance configuration object) to version 8.1, you will not be able to make cluster configuration changes until all servers with the same cluster name have been upgraded to version 8.1. If needed, you can create temporary clusters based on server versions and modify each of the servers' cluster name appropriately to minimize the impact while you are upgrading.
- The bypass-pw-policy privilege was intended to provide a way for administrators to bypass certain password restrictions that would normally be imposed when managing passwords for other users. A user with this privilege could use it to bypass password validation for their own entry. The bypass-pw-policy privilege now only applies when changing another user's password (that is, an administrative password reset), and only in the following scenarios:
  - A user with this privilege will be permitted to set pre-encoded passwords.
  - A user with this privilege will be permitted to set passwords that would otherwise be rejected by one or more password validators.
  - A user with this privilege will be permitted to set passwords that match the current password or that are in the password history.

  The privilege will only apply when changing the password for another user, and it will have no effect for a self password change. Further, a user with this privilege will no longer be exempted from any other restrictions in their own password policy. Before upgrading, you should search for users that have

the bypass-pw-policy privilege and check for compatibility issues. You can use the existing password validators and a custom password policy to enforce passwords for administrative users.

- Missing changes will now be detected when the backend is reverted and there are insufficient changes in the changelog database. When in this particular missing-changes state the local replica will not accept changes from the local replication server.
- Fixed an issue in which the server could incorrectly evaluate a matched values request control containing an extensible match filter that specified both an attribute type and a matching rule. The server incorrectly used the attribute type's equality matching rule instead of the matching rule specified in the filter.
- Updated the "delay bind response" failure lockout action to provide an option to delay the response to bind requests initiated by non-LDAP clients (for example, when using HTTP basic authentication). This option is disabled by default because delaying the bind response for non-LDAP clients may require temporarily blocking the thread used to process the request, which could increase the risk of a denial-of-service attack. To help mitigate this risk, if you enable delayed bind responses for non LDAP clients, we recommend that you also increase the number of request handler threads for all enabled HTTP connection handlers.
- Updated setup to create a second encryption settings definition if data encryption is enabled. It will continue to create a definition for 128-bit AES encryption for use as the preferred definition to preserve backward compatibility with existing servers in the topology, but it will now also generate a definition for 256-bit AES encryption. The 256-bit AES definition may become the preferred definition in a future release, but you can use it now by first ensuring that any existing instances are updated to contain the new definition (with the "encryption-settings export" and "encryption-settings import" commands) and then making it the preferred definition (with "encryption-settings set-preferred") in all instances.
- Fixed an issue that could prevent the uninstaller from removing information about the instance from the topology registry.

Critical Fixes

This release of PingDirectory Server addresses critical issues from earlier versions. Update all affected servers appropriately.

- Addressed an issue that could lead to slow, off-heap memory growth. This only occurred on servers whose cn=Version,cn=monitor entry was retrieved frequently.
  - Fixed in: 8.1.0.0
  - Introduced in: 5.2.0.0
  - Support identifiers: DS-41301
- Addressed an issue where replication could incorrectly detect a backlog that never clears when updating from a pre-7.3 to a 7.3 or later version. This issue requires that servers were previously removed from the topology, and it has been seen rarely.
  - Fixed in: 8.1.0.0
  - Introduced in: 7.3.0.0
  - Support identifiers: DS-40955
- Fixed a memory leak when performing SCIM queries on PingDirectory Server.
  - Fixed in: 8.1.0.0
  - Introduced in: 7.2.0.0
  - Support identifiers: DS-41206 SF#00681395
- Addressed an issue that could lead to slow off-heap memory growth. This only occurred on servers whose cn=Version,cn=monitor entry was retrieved frequently.
  - Fixed in: 8.0.0.1

    Introduced in: 5.2.0.0

    Support identifiers: DS-41301

- Fixed an issue that could cause the server to report an "Unable to decode a blacklist key" error while trying to open a local DB backend after an unclean shutdown.
  - Fixed in: 8.0.0.0
  - Introduced in: 7.2.0.0
  - Support identifiers: DS-40788
- The following enhancements were made to the topology manager to make it easier to diagnose connection errors:
  - Added monitoring information for all the failed outbound connections (including the time since it's been failing and the last error message seen when the failure occurred) from a server to one of its configured peers and the number of failed outbound connections.
  - Added alarms/alerts for when a server fails to connect to a peer server within a configured grace period.
    - Fixed in: 7.3.0.0
    - Introduced in: 7.0.0.0
    - Support identifiers: DS-38334 SF#00655578
  - The topology manager will now raise a mirrored-subtree-manager-connection-asymmetry alarm when a server is able to establish outbound connections to its peer servers, but those peer servers are unable to establish connections back to the server within the configured grace period. The alarm is cleared as soon as there is connection symmetry.
    - Fixed in: 7.3.0.0
    - Introduced in: 7.0.0.0
    - Support identifiers: DS-38344 SF#00655578
  - The dsreplication tool has been fixed to work when the node being used to enable replication is currently out-of-sync with the topology master.
    - Fixed in: 7.3.0.0
    - Introduced in: 7.0.0.0
    - Support identifiers: DS-38335 SF#00655578
  - Fixed the following two issues in which the server could have exposed some clear-text passwords in files on the server file system:
    - When creating an encrypted backup of the alarms, alerts, configuration, encryption settings, schema, tasks, or trust store backends, the password used to generate the encryption key (which may have been obtained from an encryption settings definition) could have been inadvertently written into the backup descriptor. This problem does not affect local DB backends (like userRoot), the LDAP changelog backend, or the replication database.
    - When running certain command-line tools with an argument instructing the tool to read a password from a file, the password contained in that file could have been written into the server's tool invocation log instead of the path to that file. Affected tools include backup, create-initial-config, create-initial-proxy-config, dsreplication, enter-lockdown-mode, export-ldif, import-ldif, ldappasswordmodify, leave-lockdown-mode, manage-tasks, manage-topology, migrate-ldap-schema, parallel-update, prepare-endpoint-server, prepare-external-server, realtime-sync, rebuild-index, re-encode-entries, reload-http-connection-handler-certificates, reload-index, remove-defunct-server, restore, rotate-log, and stop-server. Other tools are not affected. Also note that this only includes passwords contained in files that were provided as command-line arguments; passwords included in the tools.properties file, or in a file referenced from tools.properties, would not have been exposed.

    In each of these cases, the files would have been written with permissions that make their contents only accessible to the system account used to run the server. Furthermore, while administrative passwords may have been exposed in the tool invocation log, neither the passwords for regular

users, nor any other data from their entries, should have been affected. We have introduced new automated tests to help ensure that such incidents do not occur in the future.

We recommend changing any administrative passwords you fear may have been compromised as a result of this issue. If you are concerned that the passphrase for an encryption settings definition may have been exposed, then we recommend creating a new encryption settings definition that is preferred for all subsequent encryption operations, exporting your data to LDIF, and re-importing so that it will be encrypted with the new key. You also may wish to re-encrypt or destroy any existing backups, LDIF exports, or other data encrypted with a compromised key, and you may wish to sanitize or destroy any existing tool invocation log files that may contain clear-text passwords.

  - Fixed in: 7.3.0.0
  - Introduced in: 7.0.0.0
  - Support identifiers: DS-38897 DS-38908

The following enhancements were made to the topology manager to make it easier to diagnose connection errors:

- Added monitoring information for all the failed outbound connections (including the time since it's been failing and the last error message seen when the failure occurred) from a server to one of its configured peers and the number of failed outbound connections.
- Added alarms/alerts for when a server fails to connect to a peer server within a configured grace period.
  - Fixed in: 7.2.1.0
  - Introduced in: 7.0.0.0
  - Support identifiers: DS-38334 SF#00655578
- The topology manager will now raise a mirrored-subtree-manager-connection-asymmetry alarm when a server is able to establish outbound connections to its peer servers, but those peer servers are unable to establish connections back to the server within the configured grace period. The alarm is cleared when connection symmetry is achieved.
  - Fixed in: 7.2.1.0
  - Introduced in: 7.0.0.0
  - Support identifiers: DS-38344 SF#00655578
- The dsreplication tool has been fixed to work when the node being used to enable replication is currently out-of-sync with the topology master.
  - Fixed in: 7.2.1.0
  - Introduced in: 7.0.0.0
  - Support identifiers: DS-38335 SF#00655578
- Addressed an issue where an InvalidKeyException could occasionally be reported by import-ldif. The error message for this problem resembles, "An unexpected error occurred during merge processing for index 'dc_example_dc_com_sn.equality': InvalidKeyException: The provided passphrase is invalid."
  - Fixed in: 7.2.0.0
  - Introduced in: 7.0.0.0
  - Support identifiers: DS-37313
- Fixed the following two issues in which the server could have exposed some clear-text passwords in files on the server file system:
  - When creating an encrypted backup of the alarms, alerts, configuration, encryption settings, schema, tasks, or trust store backends, the password used to generate the encryption key (which may have been obtained from an encryption settings definition) could have been

inadvertently written into the backup descriptor. This problem does not affect local DB backends (like userRoot), the LDAP changelog backend, or the replication database.

- When running certain command-line tools with an argument instructing the tool to read a password from a file, the password contained in that file could have been written into the server's tool invocation log instead of the path to that file. Affected tools include backup, create-initial-config, create-initial-proxy-config, dsreplication, enter-lockdown-mode, export-ldif, import-ldif, ldappasswordmodify, leave-lockdown-mode, manage-tasks, manage-topology, migrate-ldap-schema, parallel-update, prepare-endpoint-server, prepare-external-server, realtime-sync, rebuild-index, re-encode-entries, reload-http-connection-handler-certificates, reload-index, remove-defunct-server, restore, rotate-log, and stop-server. Other tools are not affected. Also note that this only includes passwords contained in files that were provided as command-line arguments; passwords included in the tools.properties file, or in a file referenced from tools.properties, would not have been exposed.

In each of these cases, the files would have been written with permissions that make their contents only accessible to the system account used to run the server. Further, while administrative passwords may have been exposed in the tool invocation log, neither the passwords for regular users, nor any other data from their entries, should have been affected. We have introduced new automated tests to help ensure that such incidents do not occur in the future.

We recommend changing any administrative passwords you fear may have been compromised as a result of this issue. If you are concerned that the passphrase for an encryption settings definition may have been exposed, then we recommend creating a new encryption settings definition that is preferred for all subsequent encryption operations, exporting your data to LDIF, and re-importing so that it will be encrypted with the new key. You also may wish to re-encrypt or destroy any existing backups, LDIF exports, or other data encrypted with a compromised key, and you may wish to sanitize or destroy any existing tool invocation log files that may contain clear-text passwords.

- Fixed in: 7.0.1.3
- Introduced in: 7.0.0.0
- Support identifiers: DS-38897 DS-38908

- Addressed an issue in "dsreplication enable/initialize" that prevented servers from some previous versions (5.2.0.5 and earlier and 6.0.0.*) from initializing newer servers. Servers from these prior versions can now be used to enable replication with current versions of the server.
  - Fixed in: 7.0.0.0
  - Introduced in: 5.2.0.5
  - Support identifiers: DS-35528 SF#624368
- Fixed a very rare race condition with the Frequently Accessed Entry Cache which could lead to an index being marked as degraded and requiring a rebuild.

  The problem is unlikely to happen outside of testing environments since it requires modifying a single entry over 1000 times per second across multiple servers concurrently.
  - Fixed in: 7.0.0.0
  - Introduced in: 5.2.0.6
  - Support identifiers: DS-35616 SF#00625189
- Addressed an issue where an index key could incorrectly be reported as exceeding the index-entry-limit after one billion entries had been imported or added to the directory server. The directory server does not need to contain one billion entries at the same time to be affected by this issue since the entry ID will always increase for each added entry even if entries are deleted. Environments that have experienced this issue should export and reimport their data after applying this patch.
  - Fixed in: 7.0.0.0
  - Introduced in: 2.0.0.0
  - Support identifiers: DS-35790 SF#00625942

What's New

These are new features for this release of PingDirectory Server:

- The collect-support-data tool will now collect additional files if Delegated Admin is configured within PingDirectory Server. These files include the `config.js` configuration file, the version file, and any custom files used to create customized UI fields. This information can be useful when troubleshooting issues with the Delegated Admin application.
- In an ongoing effort to improve the use of containers for PingDirectory Server, several features have been implemented:

  - The --outputFile option has been added to the collect-support-data tool. You can now specify either a path, a file name, or a path and file name for the resulting CSD file. This means an administrator can run the collect-support-data tool and put the output file into a directory outside of the container, allowing access to the file without having to actually connect to the container.

    - For the **`collect-support-data`** tool, you can now specify either a path, a file name, or a path and file name for the resulting csd file. This means an administrator can run the **`collect-support-data`** tool and put the output file into a directory outside of the container, allowing access to the file without having to actually connect to the container.
    - The **`collect-support-data`** tool can now be run as a recurring task. Recurring tasks can be created using the Administration console which means that administrators do not have to connect to the container in order to run the tool.
    - The **`manage-profile`** command used for creating server profiles can also be run as a recurring task from the Administration Console. Running **`manage-profile`** as a task creates an archive file that can be downloaded with the HTTP File Servlet. This allows updated server profiles to be created from PingDirectory Server running in a container.
    - A Collect Support Tool Extended Operation has been added allowing LDAP clients to initiate the collect-support-data tool and to receive the output of the request. The LDAP SDK has been updated to support this and the --remoteServer option added to the **`collect-support-data`** tool itself can be used to send the request to another server. In other words, you can now run **`collect-support-data`** on the command line and reference another server, possibly in a container, and retrieve the output file remotely.
- The SCIMv2 REST API was added to PingDirectory Server 8.0. Now developers can "join" separate objects into one query that allows PingDirectory Server to present distinct objects in LDAP as a single SCIM Resource object. There are two general scenarios where this functionality will be used:

  - Primary and Secondary resources, where the secondary resource is represented as a field of the primary resource
  - Composite resources, where the SCIM endpoint server stitches together scattered data into a coherent resource with which the user can interact
- PingDirectory Server has a Consent REST API that allows users to create and store consents. This new feature now allows users to search for consents that have been granted to them by another party.
- PingDirectory Server now supports the ability to create composed attributes that are derived from other attributes like virtual attributes. However, composed attributes can be stored in the backend, they are persistent and can be indexed.
- To address the need for clients to obtain information about account and password policy states, a new virtual attribute Password Policy State JSON, that is a JSON object, provides read-only access to a number of account state and password policy state properties. Since the value is a JSON object, it should be easily consumable by HTTP clients using SCIM and the Directory REST API.
- PingDirectory Server currently has an HTTP file servlet extension that can be used to access files on the server filesystem. This can be used for several purposes such as serving static HTML pages. In this version, the HTTP file servlet now supports basic authentication. Users can authenticate with either a distinguished name or username. Additionally, a new file servlet will be configured automatically when creating a PingDirectory Server instance. This servlet will have as the base directory the install directory for the instance and can be used to access items such as configuration, log files, and

backups.. Directory indexing is enabled so you can browse available files. When using this file servlet, authentication is required, and access is restricted to users with the file-servlet-access privilege.

- To assist with recovering from the split-brain state the "dsreplication initialize" command will have a new "--force" option that overrides the lockdown check.
- Missing changes will now be detected when the backend is reverted and there are insufficient changes in the changelog database. While in this particular missing-changes state, the local replica will not accept changes from the local replication server.

Known Issues/Workarounds

The following are known issues in the current version of PingDirectory Server:

- Several known issues can occur when you use the Administrative Console with Tomcat 9.0.31. You can resolve these issues by upgrading to Tomcat 9.0.33 or later.
- If you use the create-systemd-script tool to create a forking systemd service, the service is stopped by the "systemctl stop ping-directory.service" command. At that time, you can see the status using the "systemctl status ping-directory.service" command. That status might contain an indication of failure: "Active: failed (Result: exit-code)". This error has to do with the way the service exits. It is harmless.

Resolved Issues

The following issues have been resolved with this release of PingDirectory Server:

Ticket ID	Description
DS-1046,DS-1204,DS-36547	Added support for remotely invoking the collect-support-data tool using an administrative task, and for invoking the tool on a regular basis as a recurring task. The tool has also been updated to add an outputPath argument to allow specifying the path or name to use for the output file.

Ticket ID	Description
DS-1103,DS-41138,DS-41956,DS-41957	Updated the password policy configuration to add support for alternate failure lockout actions, which can customize the behavior the server exhibits for an account with too many outstanding authentication failures. Available implementations include: * A "Lock Account" action that will prevent the user from authenticating or from being used as an alternate authorization identity. This matches the behavior that the server exhibited in the past when the password policy was configured to lock accounts after too many failed authentication attempts, and it remains the default behavior unless an alternative is configured. * A "Delay Bind Response" action that will allow the user to authenticate if they provide the correct credentials and their account is not otherwise unusable, but it will delay the bind response (regardless of its ultimate success or failure) by a specified duration to limit the rate at which an attacker can attempt to guess the password. The account will be permitted as an alternate authorization identity without any delay imposed for those operations. * A "No Operation" action that will not prevent or otherwise interfere with the ability to use the account to authenticate or as an alternate authorization identity, but it will still be reported as having too many failed authentication attempts for the purposes of account status notifications, the password policy state extended operation, and the password policy state JSON virtual attribute.
DS-10216	Updated the GSSAPI SASL mechanism handler to support integrity and confidentiality protection for client communication.

Ticket ID	Description
DS-11505	Changed the behavior the server exhibits for users with the bypass-pw-policy privilege. Although this privilege was primarily intended to allow administrators to be exempted from certain password policy restrictions while managing other user accounts, it also previously exempted the user from restrictions that should be imposed by their own password policy. As a result of these changes, the bypass-pw-policy privilege now only permits the following: * An account with this privilege will be permitted to set pre-encoded passwords in other user entries, regardless of the value of the allow-pre-encoded-passwords setting in the target user's password policy. This does not apply when the privileged user is changing their own password. * An account with this privilege will be permitted to set passwords in other user entries that would otherwise be rejected by one or more password validators configured in the target user's password policy. This does not apply when the privileged user is changing their own password. * An account with this privilege will be permitted to set passwords in other user entries when the new password is in the target user's password history. This does not apply when the privileged user is changing their own password. If the previous behavior (in which a privileged user is not subject to password policy restrictions for their own account) is desired, the user should be assigned a password policy that is not configured with the restrictions that are not wanted.
DS-36573	The minimum required heap size for installing PingDirectory Server has been increased to 768 MB. Other product sizes are unchanged.
DS-36726	Improved performance when adding entries that match very large composite index keys.
DS-37829	The "create-systemd-script" CLI now creates a "forking" service file since Ping services are started by a process (the "start-server" script) that is different than the actual service process.

Ticket ID	Description
DS-38122	Added support for an extended operation that can be used to invoke the collect-support-data tool from a remote system and stream the output and resulting support data archive back to the client. The collect-support-data command-line tool has been updated to support this capability through the new --useRemoteServer argument.
DS-38535	Fixed an issue that could cause the server to generate an administrative alert about an uncaught exception when trying to send data on a TLS-encrypted connection that is no longer valid.
DS-38585	Added gauge "Replication Purge Delay" in order to protect against missing changes. The new gauge will have a warning and critical threshold with respect to the effective purge delay, which can be configured by selecting "Gauges" in dsconfig.
DS-38790	Added support for a new ds-pwp-state-json virtual attribute, whose value is a JSON object that provides information about the state and configuration for the password policy with which that entry is associated.
DS-38879	Added a populate composed attribute values task that can be used to update existing entries with composed attribute values without the need to export data to LDIF and re-import. A command-line tool to invoke the task is also provided.
DS-38879	Added a "composed attribute" plugin that allows creating attributes whose values are constructed from the values of other attributes in the same entry. Its behavior is similar to that of the constructed virtual attribute, but composed values are actually stored in the entry rather than being dynamically generated at the time the entry is retrieved, and therefore may be indexed. Composed attribute values may be generated when entries are created by LDAP add operations or in an LDIF import, and their values may be updated if the source attributes are changed by modify or modify DN operations.
DS-39238	Updated the attribute value password validator to provide the ability to specify a minimum substring length when determining whether to reject a proposed password because it contains the value of another attribute in the entry.
DS-39442	Added the X-Frame-Options header in the Administrative console to prevent clickjacking attacks.

Ticket ID	Description
DS-39539,DS-41417,DS-41478	Fixed an issue where the manage-profile replace-profile subcommand was unable to create new local DB backends through dsconfig. Also fixed an issue where replace-profile could not export and re-import data from a server with multiple backends.
DS-39649,DS-40115	Optimized the searches for replication data performed by the status tool.
DS-39798	Fixed a bug in which SEMI_AGGRESSIVE and AGGRESSIVE JVM Tuning Parameters were previously allowed to both be selected.
DS-39911	Updated the character set password validator to make it possible to require that a proposed password contain characters from at least a specified minimum number of character sets. For example, you may define four optional sets containing lowercase letters, uppercase letters, numeric digits, and symbols, and then require that proposed passwords include characters from at least three of those sets.
DS-40356	Updated the manage-profile tool to prevent displaying warnings about offline config changes when starting the server.
DS-40379	Fixed an issue that could cause the load-balancing-algorithm-name configuration property of a directory server instance to be lost when joining a topology.
DS-40530	Fixed XSS vulnerabilities in the Administrative console.
DS-40532	Added a logging-error-behavior property to the log publisher, periodic stats logger plugin, and monitor history plugin configuration that can be used to specify the behavior the server should exhibit if an error occurs while attempting logging-related processing. By default, the server will preserve its previous behavior of writing a message to standard error, but it can be configured to enter lockdown mode on a logging error, in which the server will report itself as unavailable and will only accept requests from accounts with the lockdown-mode privilege and only from clients communicating over a loopback interface.
DS-40551	Fixed an issue that could prevent some tools from running properly with an encrypted tools.properties file.
DS-40567	A license is now always required when using the manage-profile replace-profile tool.

Ticket ID	Description
DS-40681	Added a cache for password policies stored in user data rather than in the configuration. The cache will hold up to 500 policies by default, but the cache size can be configured (or the cache disabled) using the maximum-user-data-password-policies-to-cache property in the global configuration.
DS-40746	Updated the logic that the server uses to select an appropriate default set of TLS cipher suites.
DS-40806	Fixed an issue that could cause the shutdown process to stall if the server is configured to use TCP to communicate with a StatsD endpoint that has become unresponsive.
DS-40817	PATCH operations on SCIM 2 for PingDirectory Server now require that the value of the schemas attribute in the request body to be "urn:ietf:params:scim:api:messages:2.0:PatchOp", in accordance with RFC 7644.
DS-40869	Made JSON the default SCIM 1.1 response content type for requests that did not specify an accept content type.
DS-40889	Fixed an issue with recurring exec tasks where the working-directory attribute was ignored.
DS-40902	Fixed an issue that could cause the server to report an error when disabling a PingOne for Customers pass-through authentication plugin.
DS-40951	Improved import-ldif indexing performance.
DS-40953	Added detection for buffer issues that could cause connections to get stuck during TLS handshake.
DS-40955	Addressed an issue where replication could incorrectly detect a backlog that never clears when updating from a pre-7.3 to a 7.3 or later version. This issue requires that servers were previously removed from the topology, and it has been seen rarely.
DS-41033	Before starting replication calculates the total backlog for each replica by adding up the outstanding changes for each remote replica. With this change obsolete replicas will no longer be included in the calculation.

Ticket ID	Description
DS-41051	Improved the logic used to determine an appropriate replication database cache size. The previous fixed size of 5MB was found to be too small in some cases, and the replication database could grow larger than expected. In deployments in which the JVM has access to at least 500MB of memory, the replication database cache will now be permitted to use up to 10f that memory. The former 5MB cache size will still be used in deployments with access to less than 500MB of memory.
DS-41054	Fixed an issue that stopped new extensions from being installed.
DS-41056	Updated the dictionary of commonly used passwords to include new values from studies released at the end of 2019.
DS-41074	Fixed an issue with the way the server reports memory usage after completing an explicitly requested garbage collection.
DS-41079	Trimming of replication changes database no longer gets stuck when the sequence number of the first change is greater than the sequence number of the last change, which can happen when "dsreplication initialize" is used to initialize a target with changes that are older than the changes the target previously had.
DS-41086	Updated the StatsD monitoring endpoint to replace any spaces, commas, or colons with underscores, and remove and single quotes or double quotes in sent metric lines. This simplifies parsing of the produced metrics.
DS-41118	A gauge called HTTP Processing (Percent) is now available. This gauge measures the server's capacity to process new incoming HTTP requests.
DS-41126	Updated the server to make the general monitor entry available to JMX clients.
DS-41142	Improved debugging support for Server SDK extensions. If debugging is enabled, the server will now generate a debug message whenever it invokes an extension. For some extension methods that return a value, the server will also generate a debug message with that return value.
DS-41206	Fixed a memory leak when performing SCIM queries on PingDirectory Server.

Ticket ID	Description
DS-41215	Updated the manage-profile replace-profile subcommand to check for encryption-related arguments in setup-arguments.txt when determining if an export and re-import of user data is necessary.
DS-41221	A new collaborator field has been added to consents. This field is for storing users with whom the consent has been shared with, in order to search consents that have been shared with a user.
DS-41235	Updated the cn=Cluster subtree to prevent clustered configuration changes when servers in the cluster have mixed versions. To make clustered configuration changes, either update all servers in the cluster to the same version, or temporarily create separate clusters by server version by changing the cluster-name property on the server instance configuration objects.
DS-41236	To avoid inconsistencies, changing clustered configuration will now require all servers in the cluster to be on the same product version. Servers will not pull any clustered configuration from the master of the cluster if they are on a different product version.
DS-41239	Missing changes are now detected for new replicas that have not yet had any changes.
DS-41252	Obsolete changes are now removed from the replication database of the target system when the target system is initialized.
DS-41261	Fixed an issue with manage-profile replace-profile where certain configuration changes for recurring task chains were not being applied.
DS-41270	Fixed an intermittent hang in dsreplication initialize due to replication failing to send an internal replication message. When the hang happens a message containing "the destination server is currently unavailable" is logged on the target of the initialize. The fix is to retry sending the internal replication message.
DS-41289	Fixed an issue that prevented password changes for topology administrators unless their password policy was configured to allow pre-encoded passwords.

Ticket ID	Description
DS-41301	Addressed an issue that could lead to slow, off-heap memory growth. This only occurred on servers whose cn=Version,cn=monitor entry was retrieved frequently.
DS-41333	Added an ssl-client-auth-policy configuration property to the HTTP connection handler to provide support for mutual TLS authentication.
DS-41366	Updated the base monitor entry to include locationName and locationDN attributes if the server is configured with a location.
DS-41396	Updated the Server SDK to add ClientContext and OperationContext methods for obtaining the name and DN of the associated client connection policy.
DS-41400	Updated the file servlet HTTP servlet extension to add support for requiring authentication in order to access the content. Access may optionally be limited to members of a specified set of groups.
DS-41441	Changed a validator error to a warning since it was deemed to be relatively harmless. The validator error was "Validator Error: The server experienced an unexpected error. Please report this problem and include this log file. Received an update requiring replication assurance, but without a location option"
DS-41465	Obsolete replicas will no longer be purged by default. To turn on the purging of obsolete replicas use "dsconfig" to set "replication-purge-obsolete-replicas" to true in the global configuration.
DS-41471	Added a global ACI that grants clients access to the pre-read and post-read request controls by default. The server will only process these controls if the requester has permission to perform the associated write operation, and the entries returned will only include attributes the requester has permission to read.
DS-41516	Added a --addBaseEntry argument to dsreplication enable. This argument can be used to add a base entry when enabling replication for an empty base DN.
DS-41622	The use-administrative-operation-request-control property is now hidden on unsupported products.

Ticket ID	Description
DS-41731	Fixed an issue that could prevent setup from generating a self-signed certificate for systems with non-ASCII hostnames.
DS-41762	Fixed an issue where mirrored subtree polling could produce config archive files that were identical or ignored the configured insignificant attributes list.
DS-41763	Updated the server behavior when returning entries in response to a search request in which the client explicitly specified a list of attributes to be returned. If any of the requested attributes exists in the entry but is not defined in the server schema, the server would have previously returned that attribute with a name formatted in all lowercase characters. The attribute will now be returned with a name that uses the same capitalization that the client used for it in the list of requested attributes.
DS-41818	Added the --zip argument to the manage-profile generate-profile subcommand, which can be used to generate a zipped server profile.
DS-41820	Added an administrative task that may be used to generate a server profile and a corresponding recurring task that may be used to invoke the task on a regular basis.
DS-41821	Added an instance root file servlet to the default configuration. HTTPS requests to /instance-root by authenticated users with the file-servlet-access privilege will be granted access to files within the server instance root.
DS-41850	Servers running on Linux will now log a warning about possible performance impacts if the current memory control group has memory.swappiness set to a nonzero value.
DS-41851	Enabled Correlated LDAP Data Views for SCIM 2 resource types on PingDirectory Server and PingDirectoryProxy Server.
DS-41939	Fixed an issue with in which the server may not generate an account status notification for the account-updated notification type for modify operations unless the operation also qualifies for other types of account status notifications.
DS-41941	Fixed an issue that prevented enabling the LDAP changelog backend with the manage-profile replace-profile tool.

Ticket ID	Description
DS-41987	Updated the PUT request in the consent service to reject requests that have duplicate collaborators to make it consistent with POST and PATCH requests.
DS-42006	The server now warns the administrator at startup if there are multiple versions of the same jar listed in the classpath, and the first one in the classpath is not the newest one.
DS-42033	Addressed an issue where some tools would throw a NullPointerException if a server was configured with a custom global result code map.
DS-42387	Updated the **`manage-profile generate-profile`** subcommand to exclude files in the `ldif/` and `bak/` directories by default when generating a server profile. If necessary, you can manually include those directories using the `--includePath` argument.

PingDirectory Server 8.0.0.5 release notes

Critical fixes

This release of the Directory Server addresses critical issues from earlier versions. Update all affected servers appropriately.

No critical issues have been identified.

Resolved issues

The following issues have been resolved with this release of the Directory Server.

Ticket ID	Description
DS-44204	Fixed an issue in which the Directory Server could incorrectly allow requests to be processed with an alternate authorization identity (for example, using the proxied authorization control, or if the requests pass through a Directory Proxy Server) whose account is in a "must change password" state. The server will now only permit the operation if it attempts to set a new password for the target user.

PingDirectory Server 8.0.0.3 release notes

Critical Fixes

This release of the PingDirectory Server addresses critical issues from earlier versions. Update all affected servers appropriately.

No critical issues have been identified

Resolved Issues

The following issues have been resolved with this release of the PingDirectory Server:

Ticket ID	Description
DS-38535	Fixed an issue that could cause the server to generate an administrative alert about an uncaught exception when trying to send data on a TLS-encrypted connection that is no longer valid.
DS-42373	Fixed an issue where unindexed LDAP searches filtered on virtual attributes sometimes omitted objectClass from the returned result.
DS-42922	Fixed a bug where restoring an incremental backup could result in the server not being able to start.

Ticket ID	Description
DS-43288	Updated setup and the replace-certificate tool to improve the way we generate self-signed certificates and certificate signing requests to make them more palatable to clients. To reduce the frequency with which administrators had to replace self-signed certificates, we previously used a very long lifetime for self-signed certificates generated by setup or the replace-certificate tool. However, some clients (especially web browsers and other HTTP clients) have started more strenuously objecting to certificates to long lifetimes, so we now generate self-signed certificates with a one-year validity period. The inter-server certificate (which is used internally within the server and does not get exposed to normal clients) is still created with a twenty-year lifetime. Also, the replace-certificate tool's interactive mode has been updated to improve the process that it uses to obtain information to include in the subject DN and subject alternative name extension for self-signed certificates and certificate signing requests. The following changes have been made in accordance with CA/Browser Forum guidelines: * When selecting the subject DN for the certificate, we listed a number of common attributes that may be used, including CN, OU, O, L, ST, and C. We previously indicated that CN attribute was recommended. We now also indicate that the O and C attributes are recommended as well. * When obtaining the list of DNS names to include in the subject alternative name extension, we previously suggested all names that we could find associated with interfaces on the local system. In many cases, we now omit non-qualified names and names that are associated with loopback interfaces. We will also warn about any attempts to add unqualified or invalid names to the list. * When obtaining the list of IP addresses to include in the subject alternative name extension, we previously suggested all addresses associated with all network interfaces on the system. We no longer suggest any IP addresses associated with loopback interfaces, and we no longer suggest any IP addresses associated in IANA-reserved ranges (for example, addresses reserved for private-use networks). The tool will now warn about attempts to add these addresses for inclusion in the subject alternative name extension.

Ticket ID	Description
DS-43480	Updated the system information monitor provider to restrict the set of environment variables that may be included. Previously, the monitor entry included information about all defined environment variables, as that information can be useful for diagnostic purposes. However, some deployments may include credentials, secret keys, or other sensitive information in environment variables, and that should not be exposed in the monitor. The server will now only include values from a predefined set of environment variables that are expected to be the most useful for troubleshooting problems, and that are not expected to contain sensitive information.
DS-43632	Fixed an issue where the "format" field is omitted from the list of operational attribute schemas in the Directory REST API.
DS-43651	The Security Guide is now available online at pingidentity.com and has been removed from the server packaging.

PingDirectory Server 8.0.0.2 Release Notes

Critical Fixes

This release of PingDirectory Server addresses critical issues from earlier versions. Update all affected servers appropriately.

No critical issues have been identified

Resolved Issues

The following issues have been resolved with this release of PingDirectory Server:

Ticket ID	Description
DS-39539,DS-41417,DS-41478	Fixed an issue where the manage-profile replace-profile subcommand was unable to create new local DB backends through dsconfig. Also fixed an issue where replace-profile could not export and re-import data from a server with multiple backends.
DS-40551	Fixed an issue that could prevent some tools from running properly with an encrypted tools.properties file.

Ticket ID	Description
DS-40828	Fixed an issue where some state associated with a JMX connection was not freed after the connection was closed. This led to a slow memory leak in servers that were monitored by an application that created a new JMX connection each polling interval.
DS-41033	Before starting replication calculates the total backlog for each replica by adding up the outstanding changes for each remote replica. With this change obsolete replicas will no longer be included in the calculation.
DS-41051	Improved the logic used to determine an appropriate replication database cache size. The previous fixed size of 5MB was found to be too small in some cases, and the replication database could grow larger than expected. In deployments in which the JVM has access to at least 500MB of memory, the replication database cache will now be permitted to use up to 1% of that memory. The former 5MB cache size will still be used in deployments with access to less than 500MB of memory.
DS-41054	Fixed an issue that stopped new extensions from being installed.
DS-41074	Fixed an issue with the way the server reports memory usage after completing an explicitly requested garbage collection.
DS-41079	Trimming of replication changes database no longer gets stuck when the sequence number of the first change is greater than the sequence number of the last change, which can happen when "dsreplication initialize" is used to initialize a target with changes that are older than the changes the target previously had.
DS-41126	Updated the server to make the general monitor entry available to JMX clients.

Ticket ID	Description
DS-41215	Updated the manage-profile replace-profile subcommand to check for encryption-related arguments in setup-arguments.txt when determining if an export and re-import of user data is necessary.
DS-41235	Updated the cn=Cluster subtree to prevent clustered configuration changes when servers in the cluster have mixed versions. To make clustered configuration changes, either update all servers in the cluster to the same version, or temporarily create separate clusters by server version by changing the cluster-name property on the server instance configuration objects.
DS-41236	To avoid inconsistencies, changing clustered configuration will now require all servers in the cluster to be on the same product version. Servers will not pull any clustered configuration from the master of the cluster if they are on a different product version.
DS-41252	Obsolete changes are now removed from the replication database of the target system when the target system is initialized.
DS-41261	Fixed an issue with manage-profile replace-profile where certain configuration changes for recurring task chains were not being applied.
DS-41289	Fixed an issue that prevented password changes for topology administrators unless their password policy was configured to allow pre-encoded passwords.
DS-41658	Fixed an issue where schema changes to a user-defined object class with one or more subordinate classes were not written to the schema file until additional schema changes were made.
DS-41939	Fixed an issue with in which the server may not generate an account status notification for the account-updated notification type for modify operations unless the operation also qualifies for other types of account status notifications.

Ticket ID	Description
DS-42609	Fixed an issue in which the Directory REST API could fail to decode certain credentials when using basic authentication.
DS-42812	Upgrade to jetty 9.4.30
DS-42815	Added a remove-attribute-type-from-schema tool that can be used to safely remove an attribute type definition from the server schema. It will ensure that the attribute type is not in use, and it will clean up metadata references to that attribute type that could have previously required re-importing the data before the attribute type could actually be removed from the schema. The remove attribute type processing can also be requested programmatically through an administrative task.
DS-43043	Fixed an issue where paged subtree searches posted to the Directory Rest API failed with error message: "Unable to decode the cookie in the simple paged results control value", whenever the search returned entries with DN length approaching or exceeding 127 characters.

PingDirectory Server 8.0.0.1 Release Notes

Critical Fixes

This release of the PingDirectory Server addresses critical issues from earlier versions. Update all affected servers appropriately.

- Fixed a memory leak when performing SCIM queries on PingDirectory Server.
  - Fixed in: 8.0.0.1

    Introduced in: 7.2.0.0

    Support identifiers: DS-41206 SF#00681395
- Addressed an issue that could lead to slow off-heap memory growth. This only occurred on servers whose cn=Version,cn=monitor entry was retrieved frequently.
  - Fixed in: 8.0.0.1

    Introduced in: 5.2.0.0

    Support identifiers: DS-41301

- Fixed an issue that could cause the server to report an "Unable to decode a blacklist key" error while trying to open a local DB backend after an unclean shutdown.
  - Fixed in: 8.0.0.0
  - Introduced in: 7.2.0.0
  - Support identifiers: DS-40788

The following enhancements were made to the topology manager to make it easier to diagnose connection errors:

- Added monitoring information for all the failed outbound connections (including the time since it's been failing and the last error message seen when the failure occurred) from a server to one of its configured peers and the number of failed outbound connections.
- Added alarms/alerts for when a server fails to connect to a peer server within a configured grace period.
  - Fixed in: 7.3.0.0
  - Introduced in: 7.0.0.0
  - Support identifiers: DS-38334 SF#00655578
- The topology manager will now raise a mirrored-subtree-manager-connection-asymmetry alarm when a server is able to establish outbound connections to its peer servers, but those peer servers are unable to establish connections back to the server within the configured grace period. The alarm is cleared as soon as there is connection symmetry.
  - Fixed in: 7.3.0.0
  - Introduced in: 7.0.0.0
  - Support identifiers: DS-38344 SF#00655578
- The dsreplication tool has been fixed to work when the node being used to enable replication is currently out-of-sync with the topology master.
  - Fixed in: 7.3.0.0
  - Introduced in: 7.0.0.0
  - Support identifiers: DS-38335 SF#00655578
- Fixed the following two issues in which the server could have exposed some clear-text passwords in files on the server file system:
  - When creating an encrypted backup of the alarms, alerts, configuration, encryption settings, schema, tasks, or trust store backends, the password used to generate the encryption key (which may have been obtained from an encryption settings definition) could have been inadvertently written into the backup descriptor. This problem does not affect local DB backends (like userRoot), the LDAP changelog backend, or the replication database.
  - When running certain command-line tools with an argument instructing the tool to read a password from a file, the password contained in that file could have been written into the server's tool invocation log instead of the path to that file. Affected tools include backup, create-initial-config, create-initial-proxy-config, dsreplication, enter-lockdown-mode, export-ldif, import-ldif, ldappasswordmodify, leave-lockdown-mode, manage-tasks, manage-topology, migrate-ldap-schema, parallel-update, prepare-endpoint-server, prepare-external-server, realtime-sync, rebuild-index, re-encode-entries, reload-http-connection-handler-certificates, reload-index, remove-defunct-server, restore, rotate-log, and stop-server. Other tools are not affected. Also note that this only includes passwords contained in files that were provided as command-line arguments; passwords included in the tools.properties file, or in a file referenced from tools.properties, would not have been exposed.

  In each of these cases, the files would have been written with permissions that make their contents only accessible to the system account used to run the server. Furthermore, while administrative passwords may have been exposed in the tool invocation log, neither the passwords for regular users, nor any

other data from their entries, should have been affected. We have introduced new automated tests to help ensure that such incidents do not occur in the future.

We recommend changing any administrative passwords you fear may have been compromised as a result of this issue. If you are concerned that the passphrase for an encryption settings definition may have been exposed, then we recommend creating a new encryption settings definition that is preferred for all subsequent encryption operations, exporting your data to LDIF, and re-importing so that it will be encrypted with the new key. You also may wish to re-encrypt or destroy any existing backups, LDIF exports, or other data encrypted with a compromised key, and you may wish to sanitize or destroy any existing tool invocation log files that may contain clear-text passwords.

- Fixed in: 7.3.0.0
- Introduced in: 7.0.0.0
- Support identifiers: DS-38897 DS-38908

The following enhancements were made to the topology manager to make it easier to diagnose connection errors:

- Added monitoring information for all the failed outbound connections (including the time since it's been failing and the last error message seen when the failure occurred) from a server to one of its configured peers and the number of failed outbound connections.
- Added alarms/alerts for when a server fails to connect to a peer server within a configured grace period.
  - Fixed in: 7.2.1.0
  - Introduced in: 7.0.0.0
  - Support identifiers: DS-38334 SF#00655578
- The topology manager will now raise a mirrored-subtree-manager-connection-asymmetry alarm when a server is able to establish outbound connections to its peer servers, but those peer servers are unable to establish connections back to the server within the configured grace period. The alarm is cleared when connection symmetry is achieved.
  - Fixed in: 7.2.1.0
  - Introduced in: 7.0.0.0
  - Support identifiers: DS-38344 SF#00655578
- The dsreplication tool has been fixed to work when the node being used to enable replication is currently out-of-sync with the topology master.
  - Fixed in: 7.2.1.0
  - Introduced in: 7.0.0.0
  - Support identifiers: DS-38335 SF#00655578
- Addressed an issue where an InvalidKeyException could occasionally be reported by import-ldif. The error message for this problem resembles, "An unexpected error occurred during merge processing for index 'dc_example_dc_com_sn.equality': InvalidKeyException: The provided passphrase is invalid."
  - Fixed in: 7.2.0.0
  - Introduced in: 7.0.0.0
  - Support identifiers: DS-37313
- Fixed the following two issues in which the server could have exposed some clear-text passwords in files on the server file system:
  - When creating an encrypted backup of the alarms, alerts, configuration, encryption settings, schema, tasks, or trust store backends, the password used to generate the encryption key (which may have been obtained from an encryption settings definition) could have been inadvertently written into the backup descriptor. This problem does not affect local DB backends (like userRoot), the LDAP changelog backend, or the replication database.
  - When running certain command-line tools with an argument instructing the tool to read a password from a file, the password contained in that file could have been written into the server's tool invocation log instead of the path to that file. Affected tools include backup, create-initial-

config, create-initial-proxy-config, dsreplication, enter-lockdown-mode, export-ldif, import-ldif, ldappasswordmodify, leave-lockdown-mode, manage-tasks, manage-topology, migrate-ldap-schema, parallel-update, prepare-endpoint-server, prepare-external-server, realtime-sync, rebuild-index, re-encode-entries, reload-http-connection-handler-certificates, reload-index, remove-defunct-server, restore, rotate-log, and stop-server. Other tools are not affected. Also note that this only includes passwords contained in files that were provided as command-line arguments; passwords included in the tools.properties file, or in a file referenced from tools.properties, would not have been exposed.

In each of these cases, the files would have been written with permissions that make their contents only accessible to the system account used to run the server. Further, while administrative passwords may have been exposed in the tool invocation log, neither the passwords for regular users, nor any other data from their entries, should have been affected. We have introduced new automated tests to help ensure that such incidents do not occur in the future.

We recommend changing any administrative passwords you fear may have been compromised as a result of this issue. If you are concerned that the passphrase for an encryption settings definition may have been exposed, then we recommend creating a new encryption settings definition that is preferred for all subsequent encryption operations, exporting your data to LDIF, and re-importing so that it will be encrypted with the new key. You also may wish to re-encrypt or destroy any existing backups, LDIF exports, or other data encrypted with a compromised key, and you may wish to sanitize or destroy any existing tool invocation log files that may contain clear-text passwords.

- Fixed in: 7.0.1.3
- Introduced in: 7.0.0.0
- Support identifiers: DS-38897 DS-38908

- Addressed an issue in "dsreplication enable/initialize" that prevented servers from some previous versions (5.2.0.5 and earlier and 6.0.0.*) from initializing newer servers. Servers from these prior versions can now be used to enable replication with current versions of the server.
  - Fixed in: 7.0.0.0
  - Introduced in: 5.2.0.5
  - Support identifiers: DS-35528 SF#624368
- Fixed a very rare race condition with the Frequently Accessed Entry Cache which could lead to an index being marked as degraded and requiring a rebuild.

  The problem is unlikely to happen outside of testing environments since it requires modifying a single entry over 1000 times per second across multiple servers concurrently.
  - Fixed in: 7.0.0.0
  - Introduced in: 5.2.0.6
  - Support identifiers: DS-35616 SF#00625189
- Addressed an issue where an index key could incorrectly be reported as exceeding the index-entry-limit after one billion entries had been imported or added to the directory server. The directory server does not need to contain one billion entries at the same time to be affected by this issue since the entry ID will always increase for each added entry even if entries are deleted. Environments that have experienced this issue should export and reimport their data after applying this patch.
  - Fixed in: 7.0.0.0
  - Introduced in: 2.0.0.0
  - Support identifiers: DS-35790 SF#00625942
- Fixed an issue that could allow users with locked accounts to change their own passwords using the password modify extended operation.
  - Fixed in: 6.2.0.0
  - Introduced in: 5.2.0.3
  - Support identifiers: DS-17074

- Addressed an issue specific to entry-balanced environments where changes received through replication are applied in the incorrect backend. This can occur if a restricted domain is disabled prior to disabling the global domain. With the restricted domain disabled, the affected server could apply the changes originally targeted for the restricted domain in the global domain. In addition, other servers in the topology will reset their generation ID for the restricted domain.
  - Fixed in: 6.2.0.0
  - Introduced in: 2.1.4.0
  - Support identifiers: DS-17237 SF#3746
- Added an alarm at warning level to notify if any of the important JVM startup arguments are missing or misconfigured.
  - Fixed in: 6.2.0.0
  - Introduced in: 5.0.0.0
  - Support identifiers: DS-12216
- Addressed an issue where a server could incorrectly report missed replication changes at startup in rare circumstances. Server A could report missed changes at startup where:
  - Server B had not received changes directly from a client for a long time (beyond the purge delay),
  - Since the last successful change, Server B had processed an operation from a client that made it deep enough in the operation processing to generate a change sequence number (CSN) but that operation was later rejected by the server,
  - Server A is shutdown, and
  - While Server A is shutdown, the Server B processes one or more changes directly from the client.
  - Fixed in: 6.2.0.0
  - Introduced in: 3.5.0.0
  - Support identifiers: DS-18035 SF#00614612
- Fixed an issue that could prevent the server from properly closing a database transaction under a sustained load of heavily conflicting write operations on a system that is processing those operations at an abnormally slow rate (for example, if the database is not cached and the disk subsystem is completely saturated).
  - Fixed in: 6.2.0.0
  - Introduced in: 6.0.1.0
  - Support identifiers: DS-18070
- Fixed an issue that could allow users with locked accounts to change their own passwords using the password modify extended operation.
  - Fixed in: 6.2.0.0
  - Introduced in: 5.2.0.3
  - Support identifiers: DS-17074
- Addressed an issue specific to entry-balanced environments where changes received through replication are applied in the incorrect backend. This can occur if a restricted domain is disabled prior to disabling the global domain. With the restricted domain disabled, the affected server could apply the changes originally targeted for the restricted domain in the global domain. In addition, other servers in the topology will reset their generation ID for the restricted domain.
  - Fixed in: 6.2.0.0
  - Introduced in: 2.1.4.0
  - Support identifiers: DS-17237 SF#3746
- Added an alarm at warning level to notify if any of the important JVM startup arguments are missing or misconfigured.
  - Fixed in: 6.2.0.0
  - Introduced in: 5.0.0.0
  - Support identifiers: DS-12216

- Addressed an issue where a server could incorrectly report missed replication changes at startup in rare circumstances. Server A could report missed changes at startup where:
  - Server B had not received changes directly from a client for a long time (beyond the purge delay),
  - Since the last successful change, Server B had processed an operation from a client that made it deep enough in the operation processing to generate a change sequence number (CSN) but that operation was later rejected by the server,
  - Server A is shutdown, and
  - While Server A is shutdown, the Server B processes one or more changes directly from the client.
  - Fixed in: 6.2.0.0
  - Introduced in: 3.5.0.0
  - Support identifiers: DS-18035 SF#00614612
- Fixed an issue that could prevent the server from properly closing a database transaction under a sustained load of heavily conflicting write operations on a system that is processing those operations at an abnormally slow rate (for example, if the database is not cached and the disk subsystem is completely saturated).
  - Fixed in: 6.2.0.0
  - Introduced in: 6.0.1.0
  - Support identifiers: DS-18070
- Fixed an issue where opening the backend database might fail with an IllegalStateException that references "exploded-index-background-deletes" when there are several backend exploded indexes.
  - Fixed in: 6.0.0.0
  - Introduced in: 4.6.0.0
  - Support identifiers: DS-15094
- The server can now detect an "out of file handles" situation on the operating system, and shut down to prevent running in an unreliable state.
  - Fixed in: 5.1.0.0
  - Introduced in: 2.1.0.0
  - Support identifiers: DS-12579 SF#2655
- Added a fail safe to the pending changes queue for the Changelog Backend that can detect and ignore recovered changes that do not need to be committed in order to prevent holding up other changes in the queue.
  - Fixed in: 5.0.0.0
  - Introduced in: 4.5.1.0
  - Support identifiers: DS-11720 SF#2453
- Disabled support for SSLv3 by default in the LDAP, HTTP, and JMX connection handlers, and for replication communication. The recently-discovered POODLE vulnerability could potentially allow a network attacker to determine the plaintext behind an SSLv3-encrypted session, which would effectively negate the primary benefit of the encryption.

  ⓘ **Note:**

  SSLv3 was initially defined in 1996, but was supplanted by the release of the TLSv1 definition in 1999 (and subsequently by TLSv1.1 in 2006 and TLSv1.2 in 2008). These newer TLS protocols are not susceptible to the POODLE vulnerability, and the server has supported them (and preferred them over SSLv3) for many years. The act of disabling SSLv3 by default should not have any adverse effect on clients that support any of the newer TLS protocols. However, if there are any legacy client applications that attempt to communicate securely but do not support the newer TLS protocols, they should be updated to support the newer protocols. In the event that there are known clients that do not support any security protocol newer than SSLv3 and that cannot be immediately updated to support a newer protocol, SSLv3 support can be re-enabled using the newly-introduced allowed-insecure-tls-protocol global configuration property. However, since communication using SSLv3 can no longer be considered

secure, it is strongly recommended that every effort be made to update all known clients still using SSLv3.

It is possible to use the server access log to identify LDAP clients that use SSLv3 to communicate with the server. Whenever an LDAP client establishes a secure connection to the server, or whenever a client uses the StartTLS extended operation to secure an existing plaintext connection, the server will generate a SECURITY-NEGOTIATION access log message. The "protocol" element of a SECURITY-NEGOTIATION access log message specifies the name of the security protocol that has been negotiated between the client and the server, and any SECURITY-NEGOTIATION messages with a protocol of "SSLv3" suggest that the associated client is vulnerable to the POODLE attack. In addition, if any connections are terminated for attempting to use the disallowed SSLv3 protocol, the access log message for that disconnect should include a message stating the reason for the termination.

  - Fixed in: 5.0.0.0
  - Introduced in: 2.1.0.0
  - Support identifiers: DS-11782
- Fixed a problem that could interfere with access to an exploded attribute index after performing an online index rebuild for that attribute.
  - Fixed in: 4.6.0.0
  - Introduced in: 4.5.1.0
  - Support identifiers: DS-10470
- Fix a bug in low level protocol buffer that could result in "uncaught exception" errors.
  - Fixed in: 4.5.0.0
  - Introduced in: 3.2.0.0
  - Support identifiers: DS-9268 SF#2002
- Improve server stability by disabling explicit garbage collections that were being caused by JMX connections.
  - Fixed in: 4.0.0.0
  - Introduced in: 3.5.0.0
  - Support identifiers: DS-7633
- Fix a bug in the LDAP Changelog where the changelog index manager could capture new changes for an attribute in one index after already hitting the end of another index. This created the possibility for changes to be missed when processing get-changelog-batch-requests at the same time that live traffic is happening.
  - Fixed in: 3.6.0.0
  - Introduced in: 3.2.0.0
  - Support identifiers: DS-7422
- Fix a bug that allows users with expired passwords to change attributes in their own entry other than password.
  - Fixed in: 3.5.0.0
  - Introduced in: 3.2.0.0
  - Support identifiers: DS-6054
- Address an issue where a directory server might resend duplicate changes when processing a GetChangelogBatch request in an environment that is under heavy load.
  - Fixed in: 3.5.0.0
  - Introduced in: 3.2.0.0
  - Support identifiers: DS-5656

- Update the PingDirectory Server to apply access controls when processing the GetAuthorizationEntryRequestControl.
  - Fixed in: 3.5.0.0
  - Introduced in: 2.0.0.0
  - Support identifiers: DS-854
- Fix a bug where PingDirectory Servers could potentially miss some update messages in large topologies after a restart.
  - Fixed in: 3.2.0.0
  - Introduced in: 3.1.0.0
  - Support identifiers: DS-3592

Resolved Issues

The following issues have been resolved with this release of the PingDirectory Server:

Ticket ID	Description
DS-40532	Added a logging-error-behavior property to the log publisher, periodic stats logger plugin, and monitor history plugin configuration that can be used to specify the behavior the server should exhibit if an error occurs while attempting logging-related processing. By default, the server will preserve its previous behavior of writing a message to standard error, but it can be configured to enter lockdown mode on a logging error, in which the server will report itself as unavailable and will only accept requests from accounts with the lockdown-mode privilege and only from clients communicating over a loopback interface.
DS-40681	Added a cache for password policies stored in user data rather than in the configuration. The cache will hold up to 500 policies by default, but the cache size can be configured (or the cache disabled) using the maximum-user-data-password-policies-to-cache property in the global configuration.
DS-40902	Fixed an issue that could cause the server to report an error when disabling a PingOne for Customers pass-through authentication plugin.
DS-40951	Improved import-ldif indexing performance.
DS-41206	Fixed a memory leak when performing SCIM queries on the PingDirectory Server.
DS-41301	Addressed an issue that could lead to slow, off-heap memory growth. This only occurred on servers whose cn=Version,cn=monitor entry was retrieved frequently.
DS-41280	The Admin Console is now deployable to an external container running Java 11 with no further downloads or updates necessary.

Ticket ID	Description
DS-41206	Fixed a memory leak when performing SCIM queries on the PingDirectory Server.

PingDirectory Server 8.0.0.0 Release Notes

Upgrade considerations

> ⓘ **Important:**
>
> If you plan to upgrade servers using a mixed-version environment where one version is earlier than 7.0 and some of the servers are still using the admin backend while others have been updated to the topology registry, do not attempt to make size changes to the topology. You cannot remove any existing servers (using `dsreplication disable`) or add new servers (using `dsreplication enable`) when in this transitional state of partially-updated servers. When a topology has been completely migrated to a 7.0 or later version with the topology registry, changes to the topology size are allowed, even in mixed-version environments (for example, mixed 7.3 and 8.3).

For other upgrade considerations, see *Upgrade overview and considerations*.

Critical Fixes

This release of the PingDirectory Server addresses critical issues from earlier versions. Update all affected servers appropriately.

- Fixed an issue that could cause the server to report an "Unable to decode a blacklist key" error while trying to open a local DB backend after an unclean shutdown.
  - Fixed in: 8.0.0.0
  - Introduced in: 7.2.0.0
  - Support identifiers: DS-40788

What's New

These are new features for this release of the PingDirectory Server:

- Improved end user experience with several feature additions to user e-mail notifications. Now administrators can configure the server to send multi-part emails, including plain text and HTML. Also, in addition to changes to account state, such as when a user account is locked, administrators can configure notifications to send whenever any configured profile attributes are changed. For example, this can be used for confirmations of self-service account changes, or as notifications for changes made by administrators.
- Better insight to server health and the performance of connected applications. Administrators can now push metrics to application insight and monitoring applications such as Splunk using two new methods. A new StatsD monitoring endpoint pushes metrics to StatsD-compatible services. Also, the periodic stats logger has been updated to use JSON-format log files, greatly simplifying the use of log forwarding tools like Splunk's Universal Forwarder or Elastic's FileBeats.
- Several password-related improvements. A new password validator has been added that rejects passwords known to have been breached or stolen, as collected by the *haveibeenpwned.com* service. Also, administrators can take advantage of a new Argon2 algorithm and password storage scheme, as well as new options to the PBKDF2 algorithm and password storage scheme.
- New SCIMv2 REST API to create, read, update, and delete (CRUD) users and other resources using JSON over HTTP. This complements the Directory REST API introduced in 7.2. While the Directory REST API is intended for custom application development, the SCIMv2 REST API is more intended for integrations with third party software and services. With SCIMv2, administrators configure the data

conventions used in PingDirectory Server – resource types that map to directory object classes, their attributes, and their locations within the Directory Information Tree (DIT).

- A new tool to easily update the security certificates used by the server. The `replace-certificate` tool allows administrators to replace the certificates presented by the server for external client connections, both HTTPS and LDAPS. If required, the replace-certificate also allows the administrator to replace the certificates used for inter-server communication, like replication.

Known Issues/Workarounds

The following are known issues in this version of the PingDirectory Server:

- The following are suggested solutions for problems with slow DNS:
  - Maintain a connection pool in the client app rather than opening new connections for each bind.
  - Add appropriate records, including PTR records, to DNS.
  - Add `options timeout:1` in the `/etc/resolv.conf` file and/or `options single-request`.
  - If IPv6 requests specifically are causing issues, add `-Djava.net.preferIPv4Stack=true` to the `start-server.java-args` line in PingDirectory Server's `config/java.properties` file, run `bin/dsjavaproperties`, and restart the server to stop the issuance of IPv6 PTR requests.
- SCIM2 PATCH operations require ACI read permissions for any required attributes in the resource's SCIM schema, even if the operation itself does not modify the required attribute itself. Otherwise, PATCH operations will fail with the error message "Applying patch ops results in an invalid resource". The following is an example of an ACI to grant read permission to the required attributes for Passthrough SCIM resource types of objectClass inetOrgPerson for a particular oauthscope:
  - `aci: (targetfilter="(objectClass=inetOrgPerson)") (targetattr="objectClass||uid||cn||givenName||sn")(version 3.0; acl "Allow read access to inetOrgPerson required attributes with OAuth scope inetOrgPersonScope"; allow (read) oauthscope="inetOrgPersonScope";)`
- Some server tools, such as `dsreplication`, `collect-support-data`, and `rebuild-index`, will fail with errors if they are run with an encrypted `tools.properties` file.
  - **Workaround:** Add the `--noPropertiesFile` argument to the server tools to prevent them from pulling information from the encrypted file.
- The working directory value used by exec tasks is not implemented for recurring exec tasks.
- Deploying the Admin Console to an external container using JDK 11 requires downloading the following dependencies and making them available at runtime (for example, by copying them to the `WEB-INF/lib` directory of the exploded WAR file).
  - groupId:jakarta.xml.bind, artifactId:jakarta.xml.bind-api, version:2.3.2
  - groupId:org.glassfish.jaxb, artifactId:jaxb-runtime, version:2.3.2

  Workaround: Deploy the Console in an external container using JDK 8.

Resolved Issues

The following issues have been resolved with this release of the PingDirectory Server:

Ticket ID	Description
DS-17278	Added a `cn=Server Status Timeline,cn=monitor` monitor entry to track a history of the local server's last 100 status changes and their timestamps. Updated the LDAP external server monitor to include attributes tracking health check state changes for external servers. The new attributes include the number of times a health check transition has occurred, timestamps of the most recent transitions, and messages associated with the most recent transitions.
DS-37042	Added an account status notification handler that can send multi-part email messages (including a plain-text part, HTML part, and optional attachments) in response to notable events affecting user accounts.
DS-37745	Added support for an `oauthscope` access control bind rule, which can be used to grant or deny access control permissions based on a set of scopes associated with an OAuth token.
DS-37859	Added the `--exportImportLocalDBData` option to the `revert-update` tool. This option will perform an export and re-import of any local db backends if there are changes that require it between the two versions. The new option does not handle the LDAP changelog and replication changelog.
DS-37881	The PingFederate Access Token Validator will now refresh its cached value of the PingFederate server's token introspection endpoint. A new attribute, `endpoint-cache-refresh`, has been added to the PingFederate Access Token Validator, which will determine how often this refresh occurs.
DS-37955	To support multiple trace loggers, each trace logger now has its own resource key, which is shown in the `Resource` column in the output of `status`. This key allows multiple alarms, due to sensitive message types for multiple trace loggers.
DS-38053	The JWT Access Token Validator no longer requires a restart after a change to one of its signing certificates.
DS-38243	Added a password validator that can use the *Pwned Passwords service (https://haveibeenpwned.com/)* to reject passwords that are known to have been compromised in data breaches.
DS-38371	Added a password storage scheme that uses the Argon2i password hashing algorithm. This scheme requires the free and open source Bouncy Castle library cryptographic, which is not included with the server. This library must be obtained from *https://bouncycastle.org/* and placed in the server `lib` directory before the storage scheme may be used.
DS-38528	Added support for a `generate password` request control that can be included in an `add` request to request that the server generate a password for the new entry. The generated password will be provided to the client in a corresponding response control.

Ticket ID	Description
DS-38560	Updated `manage-profile replace-profile` to apply configuration changes directly, when possible. If the new server profile used by `replace-profile` has changed only the `dsconfig` batch files from the original profile, then only the `dsconfig` files are applied. If no changes are detected between profiles, `replace-profile` takes no action. If changes other than `dsconfig` are detected, the full `replace-profile` process is followed.
DS-38699	Avoid sending replication messages for LDAP operations where the required associated replication information is missing.
DS-38777	Added support for updating the server version during `manage-profile replace-profile`. The server must have been originally set up with a server profile.
DS-38832	Fixed an issue that could cause the server to leak a small amount of memory each time it failed to establish an LDAP connection to another server.
DS-38863	Updated the `manage-profile setup` subcommand to set a server's cluster name to match its instance name by default. This prevents servers in the same replication topology from being in the same cluster, reducing the risk of unintentionally overwriting parts of an existing server's configuration in a DevOps environment. The `--useDefaultClusterName` argument can be used to leave the cluster name unchanged.
DS-38867	Updated the PBKDF2 password storage scheme to add support for variants that use the 256-bit, 384-bit, and 512-bit SHA-2 digest algorithms. At present, the SHA-1 variant remains the default to preserve backward compatibility with older versions. Also, in accordance with the recommendations in NIST SP 800-63B, we have increased the default iteration count from 4096 to 10,000, and the default salt length from 64 bits to 128 bits.
DS-38869	Updated the `remove-defunct-server` tool's `--ignoreOnline` option. When using `--ignoreOnline` in a mixed-version environment, all servers must support the option.
DS-39099	Fixed an issue where error log notifications were generated regardless of whether the account status notification handler has the property enabled or disabled.
DS-39176, DS-39308	Updated the Groovy scripting language version to 2.5.7. For a list of changes, visit *groovy-lang.org* and view the Groovy 2.5 release notes. As of this release, only the core Groovy runtime and the groovy-json module are bundled with the server. To deploy a Groovy-scripted Server SDK extension that requires a Groovy module not bundled with the server, such as groovy-xml or groovy-sql, download the appropriate jar file from *groovy-lang.org* and place it in the server's `lib/extensions` directory.

Ticket ID	Description
DS-39253	Added a `replace-certificate` tool, which can help an administrator replace the listener or inter-server certificate for a server instance.
DS-39325	Removed the legacy product-specific scripts for starting and stopping the server. These include: ▪ `start-ds` and `stop-ds` for PingDirectory Server ▪ `start-proxy` and `stop-proxy` for PingDirectoryProxy Server ▪ `start-sync` and `stop-sync` for Data Sync Server ▪ `start-metrics-engine` and `stop-metrics-engine` for Data Metrics Server These legacy scripts had been deprecated for several releases in favor of the more `general start-server` and `stop-server` scripts, and they displayed a warning message about their pending removal if they were invoked. If you still have dependencies on these legacy product-specific scripts, you will need to update them to reference the general start-server and stop-server scripts instead. If it is not feasible to update these references immediately, you may create symbolic links that use the legacy script names and point at the start-server and stop-server scripts.
DS-39347	Fixed an issue where Delegated Admin would not work properly if the name of the REST Resource Type was not the same as the resource endpoint.
DS-39373	Preserve the privileges that are explicitly set on the admin user when migrating from the admin backend to the topology registry.
DS-39518	Fixed an issue in which escaped characters in schema extensions may not be handled properly. If used in attribute type constraints (such as `X-VALUE-REGEX`), this could cause unexpected or incorrect behavior.
DS-39525, DS-39526	Delegated Admin enhancements for constructed attributes. ▪ Allow a required attribute to be read-only if it is constructed. ▪ Add a configured list of "Update Constructed Attributes" on the REST resource type, similar to the "Post Create Constructed Attributes", so that constructed attributes can be updated when dependent attributes change. ▪ Handle constructed attributes which reference other constructed attributes.
DS-39589	Fixed an issue that could cause access log messages for bind and StartTLS results to report the client connection policy that had previously been assigned to the connection rather than the new policy that is in place as a result of the operation.
DS-39592	HTTP External Servers have a new attribute, `ssl-cert-nickname`, which defines the alias of a specific certificate within their keystore to be used as a client certificate.

Ticket ID	Description
DS-39603	Fixed an issue where Server SDK extensions could not be configured by `dsconfig` batch files in the manage-profile tool.
DS-39626, DS-40357	The trace log publisher will now record an access token's scopes after the token is successfully validated.
DS-39654	Added support for the `--topologyFilePath` argument to the `manage-topology add-server` subcommand.
DS-39671	Updated the `manage-topology add-server` subcommand to require being run from the older server in a mixed-version environment.
DS-39693	Fixed an issue where Delegated Admin search results were truncated and invalid upon encountering a Directory entry containing a Boolean or Integer syntax attribute whose values were invalid because they did not conform to the appropriate syntax. With this fix, the offending values are omitted from the results and a warning message is logged to the server errors log.
DS-39715	Updated the Server SDK to add support for sending email messages.
DS-39762	Added support for a "generate password" extended operation that can be used to request that the server generate one or more passwords that may be suggested as possible values when creating a new user or changing the password for an existing user.
DS-39787	Fix verify-index errors on multibyte UTF-8 strings.
DS-39796	Added support for two new account status notification types. The account-created notification type can be used to generate a notification whenever a new entry is created in an add request that matches a given set of criteria. The account-updated notification type can be used to generate a notification whenever an entry is updated in a modify or modify DN request that matches a given set of criteria.
DS-39797	Updated the account status notification handler framework to make several new properties available for use in constructing notification messages. These properties provide information about the password policy configuration and the user's password policy state, including: ▪ The time the notification was generated ▪ The DN of the account with which the notification is associated ▪ The time the account password was last changed ▪ Whether the account is in a usable state and information about any issues that might affect its usability ▪ Whether the account is administratively disabled ▪ Whether the account is expired or not yet active ▪ Information about any account lockout that may be in effect ▪ Information about password expiration ▪ Whether a new password was generated for a self-change or administrative reset

Ticket ID	Description
DS-39857	Added the StatsD monitoring endpoint. When the Stats Collector Plugin is enabled, this endpoint sends metric data from the server in StatsD format to the configured destination.
DS-39872	Updated Summarize Access Log Rotation Listener and Copy Log File Rotation Listener to run on a background thread.
DS-39873	Fixed an issue that allowed replicated subtree deletes to cause OutOfMemory errors on replicas. Also fixed a related issue that would cause the replication log to fill up with mild errors.
DS-39882	Fixed an issue that could prevent the simple request criteria from properly evaluating a target entry filter or group membership for a modify DN operation after the change had already been applied to the backend.
DS-39908	Added a new JVM-default trust manager provider that can be used to automatically trust any certificate signed by an authority included in the JVM's default set of trusted issuers. Also, updated other trust manager providers to offer an option to use the JVM-default trust addition to the trust that they normally provide.
DS-40114	Added a new `cn=Status Health Summary,cn=monitor` monitor entry that provides a summary of the server's current assessment of its health. This simplifies monitoring with third party tools that support retrieving monitoring data over JMX. The Periodic Stats Logger has also been updated to allow some of this monitoring information to be logged. No new information is logged by default.
DS-40154	Fixed an issue where the restore tool was not restoring all dependencies of an incremental backup.
DS-40177	Fixed an issue that could interfere with the server's ability to purge an earlier, full backup when the following conditions are satisfied: Retention is used. Multiple levels of incremental backups depend on the full backup."
DS-40210	Fixed an issue that prevented uniqueMembers from adding themselves to a groupOfUniqueNames.
DS-40249	Fix an issue where an LDAP search across entry-balanced server sets sometimes returned 0 (success) even though all servers in one of the sets failed with a timeout. The search should return 52 (unavailable) in this situation.
DS-40317	Created a new function that takes index objects rather than the counts so that we can avoid adding to a `long.max_value`, creating a long overflow.
DS-40354	Fixed a problem with `config-diff` when writing properties that span multiple lines using the `--prettyPrint` argument.
DS-40366	Fixed an issue where the server was attempting to connect by an IP address rather than a hostname when DNS lookup was successful.
DS-40377	Added support for logging to a JSON file in the Periodic Stats Logger Plugin.

Ticket ID	Description
DS-40409	Updated the `ldif-diff` tool so that it provides a `--stripTrailingSpaces` option that can cause the LDIF parser to strip off illegal trailing spaces rather than reject the associated entry or change record.
DS-40517	Added metrics for status summary, replication database, and LDAP changelog to the Stats Collector Plugin.
DS-40543	Updated `manage-profile replace-profile` to copy the tool log file to the server being updated.
DS-40556	Added support for specifying a working directory for exec tasks.
DS-40561	Fixed an issue that prevented assured replication from working for requests received via SCIM or the REST API.
DS-40674	Added the `--addMissingRdnAttributes` argument to `manage-profile` setup. This argument can be used when including LDIF files in the server profile. It will automatically add any missing RDN values to the set of entry attributes when they are not already present.
DS-40730	Updated the `encrypt-file` tool to prevent using the same path for both the input file and the output file.
DS-40744	Fixed an issue with the interaction between deprecated password storage schemes and forced password reset. If a user's password is reset by an administrator using a password storage scheme that is subsequently configured as deprecated, the act of re-encoding the password with the new default scheme would have incorrectly cleared the password reset flag.
DS-40771	Added a `--duration` argument to collect-support-data. When used, only the log files covering the specified duration before the current time will be collected.
DS-40788	Fixed an issue that could cause the server to report an "Unable to decode a blacklist key" error while trying to open a local DB backend after an unclean shutdown.
DS-40799	Fixed an issue in which an account that had been temporarily locked after too many failed authentication attempts could become re-locked with fewer than the expected number of subsequent failed attempts after the previous lockout period had elapsed.

PingDirectoryProxy Server Release Notes

PingDirectoryProxy 8.2.0.8 release notes

The release notes for the 8.2.0.8 release of the PingDirectoryProxy server.

Critical fixes

This release of the PingDirectoryProxy server addresses critical issues from earlier versions. Update all affected servers appropriately.

No critical issues have been identified.

Resolved issues

The following issues have been resolved with this release of the PingDirectoryProxy server.

Ticket ID	Description
DS-45164	Updated Jetty Server to version 9.4.44.
DS-45813	Updated PingDirectory products to use Kafka v2.8.1.

PingDirectoryProxy Server 8.2.0.7 release notes

The release notes for the 8.2.0.7 release of PingDirectoryProxy Server.

Critical fixes

This release of PingDirectoryProxy Server addresses critical issues from earlier versions. Update all affected servers appropriately.

- Fixed an issue where secret keys under `cn=Topology,cn=config` could be lost when removing a server from the topology. When a server is removed via the `dsreplication disable` or **`remove-defunct-server`** tools, its secret keys will now be distributed among the remaining members of the topology. The keys from the rest of the topology will also be copied to the server being removed.

  The cipher secret keys in the topology that are affected by this change are used by reversible password storage schemes (except for AES256, which uses the encryption settings database). If you are using a reversible password storage scheme other than AES256, prior to this fix, you could lose access to keys that had been used for reversible password encryption when removing servers from the topology.

  ⓘ **Note:**

  Since this change only applies to the most recent version of **`remove-defunct-server`** and `dsreplication disable`, if you are removing a server from a multi-version topology, you should run that tool from the most recent version. In the past **`dsreplication`** and **`remove-defunct-server`** could only be run from an older version, but now in the case of removing a server from the topology, they should be run from the most recent version in the topology. If you run the tool from an older server, it will not include this fix, and you might lose access to secret keys from servers that are removed from the topology.

  - Fixed in: 8.2.0.7
  - Introduced in: 7.0.0.0
  - Support identifiers: DS-44591

Resolved Issues

The following issues have been resolved with this release of the Directory Proxy Server.

Ticket ID	Description
DS-44591	Fixed an issue where secret keys under `cn=Topology,cn=config` could be lost when removing a server from the topology. When a server is removed via the `dsreplication disable` or **`remove-defunct-server`** tools, its secret keys will now be distributed among the remaining members of the topology. The keys from the rest of the topology will also be copied to the server being removed. The cipher secret keys in the topology that are affected by this change are used by reversible password storage schemes (except for AES256, which uses the encryption settings database). If you are using a reversible password storage scheme other than AES256, prior to this fix, you could lose access to keys that had been used for reversible password encryption when removing servers from the topology. ⓘ **Note:** Since this change only applies to the most recent version of **`remove-defunct-server`** and `dsreplication disable`, if you are removing a server from a multi-version topology, you should run that tool from the most recent version. In the past **`dsreplication`** and **`remove-defunct-server`** could only be run from an older version, but now in the case of removing a server from the topology, they should be run from the most recent version in the topology. If you run the tool from an older server, it will not include this fix, and you might lose access to secret keys from servers that are removed from the topology.
DS-45124	Removed `-XX:RefDiscoveryPolicy=1` from the default start-server Java arguments. In rare cases, this argument was related to segmentation faults in the JVM, especially when used with the G1 garbage collector.
DS-45190	Added support for the use of JDKs obtained through BellSoft.

PingDirectoryProxy Server 8.2.0.6 release notes

Critical fixes

This release of PingDirectoryProxy Server addresses critical issues from earlier versions. Update all affected servers appropriately.

No critical issues have been identified.

Resolved issues

The following issues have been resolved with this release of the DirectoryProxy Server.

Ticket ID	Description
DS-36257, DS-43909	Updated the logic the server uses to maintain the `entry-balancing` global index. Improvements include: ▪ In some cases, the server could have incorrectly populated attribute indexes in the course of processing an **`add`** operation. This could have led to scenarios in which the global index had incomplete information for some attribute values, which could have caused the Directory Proxy Server to only return entries from a subset of the backend sets that contained those entries. Global attribute indexes now provide a `guaranteed-unique` configuration property, and they will now only be populated with new records after an **`add`** operation for indexes that are marked as guaranteed unique. ▪ Global attribute indexes were previously not updated for **`add`** operations that contained an indexed attribute in the relative distinguished name (RDN) but that attribute was not included in the set of attributes for the **`add`** operation. The appropriate index updates will now be performed for the operation. ▪ The global index was previously not updated for operations that were part of a successful transaction or **`multi-update`** extended operation. The appropriate updates will now be made. ▪ Global attribute indexes were previously not updated as a result of modifications used to remove specific attribute values from an entry. Indexes that are marked as guaranteed unique will now be updated for delete modifications that list the values to remove from the entry. ▪ Global attribute indexes were previously not updated for **`delete`** operations. The server has been updated so that if the entry's RDN includes attributes for which there are indexes marked as guaranteed unique, the index will be updated to remove those RDN values.
DS-43632	Fixed an issue where the `format` field is omitted from the list of operational attribute schemas in the Directory REST API.

PingDirectoryProxy Server 8.2.0.5 release notes

Critical fixes

This release of the Directory Proxy Server addresses critical issues from earlier versions. Update all affected servers appropriately.

No critical issues have been identified.

Resolved issues

The following issues have been resolved with this release of the Directory Proxy Server.

Ticket ID	Description
DS-44025	Fixed an issue where the server was incorrectly displaying an `Unknown vendor` warning when using JDKs obtained on Red Hat and Ubuntu systems.
DS-44204	Fixed an issue in which the Directory Server could incorrectly allow requests to be processed with an alternate authorization identity (for example, using the proxied authorization control, or if the requests pass through a Directory Proxy Server) whose account is in a "must change password" state. The server will now only permit the operation if it attempts to set a new password for the target user.

PingDirectoryProxy Server 8.2.0.2 release notes

Critical Fixes

This release of the PingDirectoryProxy Server addresses critical issues from earlier versions. Update all affected servers appropriately.

No critical issues have been identified

Resolved Issues

The following issues have been resolved with this release of the PingDirectoryProxy Server:

Ticket ID	Description
DS-43224	Made a generic OpenID Connect ID token validator available. This change allows single sign-on to the Administrative Console with OIDC providers other than just PingOne.
DS-43939	Fixed an issue where PingDirectoryProxy Server sometimes returns incorrect result code 80 ("Other") or 81 ("Server down") instead of 32 ("No Such Entry") when adding child entries with missing parents to an entry-balanced set with only one backend server.

Ticket ID	Description
DS-43941	You can now specify that the Administrative Console use a custom truststore when evaluating OIDC provider certificates by using the oidc-trust-store-file and oidc-trust-store-type settings. Also, you can set the console to skip hostname and/or certification verification through the oidc-strict-hostname-verification and oidc-trust-all configuration settings.

PingDirectoryProxy Server 8.2.0.1 Release Notes

Critical Fixes

Resolved Issues

This release of the PingDirectoryProxy Server addresses critical issues from earlier versions. Update all affected servers appropriately.

No critical issues have been identified

The following issues have been resolved with this release of the PingDirectoryProxy Server:

Ticket ID	Description
DS-43745	Fix an issue in the server causing PingDirectory Server 8.2.0.0 to be incompatible with version 4.2.0 of the Delegated Admin app, which prevented upgrade to Delegated Admin version 4.4.0 without some downtime.

PingDirectoryProxy Server 8.2 release notes

Enhancements

The following is a new feature for this release of PingDirectoryProxy Server:

- With SCIMv2, client applications might need to perform a sorted search on a large dataset. In this version of PingDirectory, SCIMv2 searches with the proper search parameters come back in pages.
- The Administrative Console now supports using OpenID Connect for admin SSO, allowing you to set up the PingOne administration console to have one-click SSO access without typing a password.

Upgrade considerations

Upgrade considerations are no longer part of the release notes. That information is now in *Upgrade overview and considerations* on page 891.

Known issues and limitations

The following are known issues in the current version of PingDirectoryProxy Server:

When signing on to the Administrative Console, you must specify an LDAPS port in the **Server** field. For example, to use port 636, you would specify pingdirectory:636. If you do not specify a port, you get a

```
"Server
      unavailable"
```

message.

Resolved Issues

The following issues have been resolved with this release of PingDirectoryProxy Server:

Ticket ID	Description
DS-644, DS-42031	Added a new validate-ldap-schema tool that can be used to examine schema definitions in a set of LDIF files and report any issues that it detects.
DS-5040	Improved the **`dsframework`** tool to support multivalued server properties.
DS-5143, DS-11035	Updated support for logging access and error log messages to a syslog server. While the server previously supported logging these messages to a syslog server (through the "syslog-based access log publisher" and "syslog-based error log publisher" logger implementations), these loggers used an older version of the syslog protocol (described in RFC 3164) and only offered support for communicating over UDP. These loggers are still available for legacy backward compatibility, but we now also offer new "syslog text access log publisher" and "syslog text error log publisher" implementations that use a newer version of the syslog protocol (syslog version 1, described in RFC 5424) and support communicating over UDP or the more reliable TCP. When using TCP, it is also possible to encrypt communication with TLS, and it is possible to configure multiple servers for better redundancy. These loggers use the same space-delimited text format as the former loggers. We also offer new "syslog JSON access log publisher" and "syslog JSON error log publisher" implementations that offer the same set of capabilities, but that format the message text as JSON objects, which can be more easily parsed by third-party software.

Ticket ID	Description
DS-7475, DS-7605, DS-11725, DS-37707, DS-40342, DS-41940	Made several improvements to the parallel-update tool, which can use multiple concurrent threads for improved performance when applying add, delete, modify, and modify DN changes read from an LDIF file. The enhancements include: * Added support for several additional controls, including proxied authorization, manageDsaIT, ignore NO-USER-MODIFICATION, password update behavior, operation purpose, name with entryUUID, assured replication, replication repair, suppress operational attribute update, and suppress referential integrity updates. The tool also now supports specifying arbitrary controls for inclusion in add, bind, delete, modify, and modify DN requests. * Made its communication more robust. The tool would previously establish connections only when it was first started, but it can now detect when a connection is no longer valid and can re-establish connections as needed to continue processing. Further, if an operation failed because it was attempted on an invalid connection, that operation can be automatically retried immediately on a newly established connection. * Added support for failover directory servers. You can now provide the --hostname and --port arguments multiple times to specify information about multiple directory server instances. In the event that the first server listed is not available (or becomes unavailable in the middle of processing), it can automatically try establishing a connection to an alternative server to continue processing there. * Improved the ability to determine whether all changes were processed successfully. Previously, the tool would always use an exit code of zero if it was able to attempt all of the changes read from the LDIF file, even if some of the changes were not successfully applied. That is still the default behavior, but a new --useFirstRejectResultCodeAsExitCode argument can be used to indicate that if any operations are rejected, the result code from the first rejected operation should be used as the exit code. * Added support for encrypted LDIF files. If the LDIF file was encrypted with a key from the server's encryption settings database, then the tool will automatically attempt to retrieve the appropriate key. Otherwise, the new --encryptionPassphraseFile argument can be used to supply the encryption passphrase, or the passphrase can be interactively requested from the user. * The tool is now parallel by default. Previously, if you did not specify a value for the --numThreads argument, it would use a single thread. It now defaults to using eight threads. * The tool now provides a --followReferrals argument that allows it to automatically attempt to follow any referrals that are returned. * The tool now provides a menu-driven interactive mode that can be used to provide values for all of the command-line arguments.
DS-10320, DS-12550, DS-12551, DS-12552, DS-42116, DS-42162, DS-42179, DS-42222, DS-42223, DS-42224, DS-42225, DS-42416, DS-42437	Added a config/sample-dsconfig-batch-files directory with set of well commented dsconfig batch files that may be useful in enabling or configuring a variety of features in the server.

Ticket ID	Description
DS-10775	Updated the dictionary password validator to support additional options: * It can now ignore non-alphabetic characters that appear at the beginning or end of the password before checking the dictionary. * It can strip characters of diacritical marks, including accents, cedillas, circumflexes, diaereses, tildes, and umlauts, before checking the dictionary. If this option is used, then any character with such a mark will be replaced with a base version of the character without that mark (for example, a lowercase letter n with a tilde over it would be replaced with just a lowercase letter n). * You can define maps with information about character substitutions to use for checking alternative versions of the provided password. For example, if you indicate that "0" might map to "o", "1" or "!" might map to "i", "7" might map to "t", and "3" might map to "e", then the validator can reject a proposed password of "pr0h1b!73d" if the dictionary contains the word "prohibited". * It can reject a proposed password if a value from the provided dictionary makes up more than a specified percentage of that password.
DS-11524, DS-41860, DS-42112	Added support for new administrative alert types: * We have added a new admin alert account status notification handler, which can generate administrative alerts whenever an applicable account status notification is generated within the server. For example, this account status notification handler can be added to the root password policy to generate an alert whenever a root user's password is updated or their account is locked as a result of too many failed authentication attempts. A separate alert type has been defined for each account status notification type. * We have added a new "privilege-assigned" administrative alert that can be raised whenever a new entry is added or an existing entry is updated to include one or more privileges. * We have added a new "insecure-request-rejected" administrative alert that can be raised whenever the server rejects a request as a result of the reject-insecure-requests global configuration property.
DS-12143	Updated the PingDirectoryProxy Server to improve support for the PLAIN, UNBOUNDID-DELIVERED-OTP, UNBOUNDID-TOTP, and UNBOUNDID-YUBIKEY-OTP SASL mechanisms. Previously, the PingDirectoryProxy Server itself performed all of the processing for those SASL mechanisms and would only work if the PingDirectoryProxy Server could retrieve the appropriate encoded credentials from the backend Directory Server. It will now forward the bind request to the backend server for processing, which allows it to work in deployments in which the backend server prevents the PingDirectoryProxy Server from accessing the stored credentials.
DS-13853	Added support for the OAUTHBEARER SASL mechanism (as described in RFC 7628) to allow LDAP clients to authenticate with OAuth 2.0 bearer tokens.
DS-15848, DS-42360	Added support for invoking a specified set of password validators during bind operations. If the password used to authenticate fails to satisfy one or more of the configured validators, the bind attempt can be rejected, the user can be forced to change their password, or the server can generate an account status notification to take some alternative action (for example, notifying the end user or server administrators).
DS-15864	Replaced the ldappasswordmodify tool with a new version that offers more functionality, including support for additional controls, support for multiple password change methods (the password modify extended operation, a regular LDAP modify operation, or an Active Directory-specific modify operation), and the ability to generate the new password on the client.

Ticket ID	Description
DS-17903	Updated setup to provide a --populateToolPropertiesFile argument that will allow it to populate the config/tools.properties file with default values for command-line tool arguments. If requested, properties will be provided for the server address, port, and communication security, and may also include a default bind DN and optionally a bind password. When running setup interactively, it will now prompt to determine which properties (if any) should be populated in the properties file.
DS-36088	Updated the crypto manager to make it possible to augment the set of enabled TLS cipher suites with specific suites to add to or remove from the default set of enabled suites. To enable one or more suites in addition to those in the default set, prefix the names of those suites with the "+" symbol. To disable one or more suites in the default set of enabled suites, prefix the names of those suites with the "-" symbol. This was already possible when configuring cipher suites for the LDAP and HTTP connection handlers, but it was not an option for the crypto manager.
DS-38110	Updated the System Information monitor with an "isDocker" attribute to identify if the server is running in a Docker container.
DS-38118, DS-42495	Made several updates related to the server's handling of data written to standard output and standard error: * The server can now be configured to rotate the logs/server.out file once it reaches a given size, and it will retain a configurable number of those log files. By default, the server will rotate the file once it reaches 100 megabytes and will keep up to ten files. * To better facilitate capturing log data in containerized environments, the server now supports writing JSON-formatted access and error log messages to the JVM's original standard output and error streams (which will be separate from the server.out file when the server is started with the --nodetach argument). * It is now possible to prevent the server from logging messages during startup in non-JSON format. It is also possible to prevent messages about administrative alerts from being written to standard error, or to write those messages in JSON format. These options are especially useful when using JSON-based logging to the console in no-detach mode, as they can help ensure that everything written to standard output and standard error will be formatted as JSON objects.
DS-38816, DS-41995	Updated support for the uniqueness request control to provide a more reliable mechanism for preventing conflicts that arise from operations processed concurrently in the same or different servers. If indicated in the request control, a temporary conflict prevention details entry can be added to the server before searching for existing conflicts, and that entry can be detected during pre-commit processing for other operations with uniqueness request controls that attempt to make a conflicting change. The server has also been updated to make it possible to generate an administrative alert if a uniqueness conflict is detected during post-commit processing for the uniqueness request control. Even though the conflict cannot be prevented at this stage in processing, the alert can let administrators know about it as soon as it happens so they can take any appropriate corrective action.

Ticket ID	Description
DS-38868	Updated setup to create a second encryption settings definition if data encryption is enabled. It will continue to create a definition for 128-bit AES encryption for use as the preferred definition to preserve backward compatibility with existing servers in the topology, but it will now also generate a definition for 256-bit AES encryption. The 256-bit AES definition may become the preferred definition in a future release, but you can use it now by first ensuring that any existing instances are updated to contain the new definition (with the "encryption-settings export" and "encryption-settings import" commands) and then making it the preferred definition (with "encryption-settings set-preferred") in all instances.
DS-39789	Updated the JVM memory usage monitor provider to fix an issue that could prevent the monitor from reporting the total amount of memory held by all memory consumers. Also, fixed an issue that could cause the memory-consumer attribute to use an incomplete message for consumers without a defined maximum size and added an additional memory-consumer-json attribute whose values are JSON objects with data that can be more easily extracted by automated processes.
DS-40650	Updated the collect-support-data tool to make it possible to specify how much data should be captured from the beginning and end of each log file to include in the support data archive. You can also specify the capture size when invoking the tool through an administrative task, recurring task, or extended operation.
DS-40828	Fixed an issue where some state associated with a JMX connection was not freed after the connection was closed. This led to a slow memory leak in servers that were monitored by an application that created a new JMX connection each polling interval.
DS-40967	Eliminated a misleading error message that could be logged at startup if the server was configured with one or more ACIs that only apply when using specific SASL mechanisms.
DS-41350	Fixed an issue where disabling certain backends (such as 'alarms') caused an internal monitor to log unnecessary error messages every few seconds, about not being able to gather data from that backend. Note that deliberately disabling the 'alarms' backend is not recommended in normal operation, but may occur during backup/restore operations.
DS-41521	The SCIM 2 service on PingDirectory and PingDirectoryProxy now automatically generates a Swagger 2 specification document based on the server's SCIM 2 configuration. To view this document, go to https://<your-server>/api-docs in a web browser.
DS-41865	PingDirectory and PingDirectoryProxy can now perform SCIM 2 paged searches on result sets greater than the configured lookthrough limit. To perform such a search, you must first configure a virtual list view (VLV) index.
DS-41964	Fixed an issue with the manage-profile tool where files in a server profile's dsconfig/ directory without a ".dsconfig" extension could cause failures in manage-profile replace-profile when validating updated dsconfig files.
DS-41989	Fixed an issue that could result in duplicate column headers being produced by the Periodic Stats Logger, even when the header-prefix-per-column attribute was set to true.
DS-42045	Updated the Stats Collector Plugin with a new generate-collector-files configuration property. When using the plugin exclusively for providing metrics to one or more StatsD Monitoring Endpoints, set this property to false to prevent unnecessary I/O.

Ticket ID	Description
DS-42059, DS-42060	Updated setup to add options for improving communication security: * Non-interactive setup now offers a --rejectInsecureRequests argument that will configure the server to reject any request received over a connection that is not encrypted with SSL or StartTLS. * Non-interactive setup now offers a --rejectUnauthenticatedRequests argument that will configure the server to reject any request received over a connection that is not authenticated (or that is authenticated as the anonymous user). * Interactive setup now allows you to configure the server with the LDAP connection handler disabled (which was already an option when using non-interactive setup), or enabled but only for communication encrypted with StartTLS. The --rejectInsecureRequests and --rejectUnauthenticatedRequests arguments can also be used with manage-profile by including them in the setup-arguments.txt file of the server profile.
DS-42061	Updated the interactive command-line tool framework to prefer establishing secure LDAP connections over insecure connections. Previously, when prompting for the information needed to establish a connection, the default option was to create an unencrypted LDAP connection. Now, tools will default to creating an SSL-encrypted connection if the server supports it, or to creating a StartTLS-encrypted connection if that is available but SSL is not. Tools will also default to using streamlined settings when establishing secure connections. Previously, they would always prompt about how to determine whether the server's certificate chain should be trusted. When using the streamlined settings, the tools will only prompt about certificates that cannot automatically be considered trusted using information in the JVM's default trust store, the server's default trust store (config/truststore), or the server's topology registry.
DS-42062	Updated the root password policy so that LDAP bind responses for root users and topology administrators will be delayed by one second after five consecutive failed authentication attempts.
DS-42063	Updated the "delay bind response" failure lockout action to provide an option to delay the response to bind requests initiated by non-LDAP clients (for example, when using HTTP basic authentication). This option is disabled by default because delaying the bind response for non-LDAP clients may require temporarily blocking the thread used to process the request, which could increase the risk of a denial-of-service attack. To help mitigate this risk, if you enable delayed bind responses for non-LDAP clients, we recommend that you also increase the number of request handler threads for all enabled HTTP connection handlers.
DS-42115	Updated the server's command-line tool framework to make it easier and more convenient to communicate with the server over a secure connection when no trust-related arguments are provided. Most non-interactive tools will now check the server's default trust store, the topology registry, and the JVM's default trust store to see if the presented certificate chain can be automatically trusted without the need to prompt the user. If the presented chain cannot be automatically trusted, the user may be interactively prompted to determine whether it should be trusted.

Ticket ID	Description
DS-42157, DS-43033	Updated the CRAM-MD5 and DIGEST-MD5 mechanism handlers so that they are no longer considered secure. Although the credentials are encoded in transit, their protection relies on the weak MD5 digest. Further, they require that user passwords be encoded in a reversible form so the server can retrieve them in the clear for use in authentication processing, which increases the risk that they will be exposed in a data breach. This primarily affects the ability to use these SASL mechanisms over an unencrypted connection for users that are required to authenticate in a secure manner (for example, if their password policy has require-secure-authentication set to true, or if their entry has a ds-auth-require-secure-authentication operational attribute with a value of true). These SASL mechanisms are still enabled by default for legacy backward compatibility purposes, but we discourage their use. To assist with that, we have also provided a sample dsconfig batch file that can be used to disable these SASL mechanism handlers.
DS-42199	Optimized some searches commonly used by the status tool. This should improve the performance of the tool in more complex or large-scale environments.
DS-42265	Upgrade to jetty 9.4
DS-42276	Fixed an issue where using the encryption-settings tool to import definitions with the set-preferred flag could result in none of the imported definitions being set as the preferred definition.
DS-42279	Updated the server to require a minimum key size of 2048 bits when negotiating a TLS cipher suite that uses ephemeral Diffie-Hellman key exchange.
DS-42298	Replaced the ldifsearch, ldifmodify, and ldif-diff command-line tools with more full-featured and robust implementations.
DS-42331	Replaced the ldapcompare tool with a new version that offers more functionality, including support for multiple compare assertions, following referrals, additional controls, and multiple output formats (including tab-delimited text, CSV, and JSON).
DS-42347	Updated the server to use /dev/urandom (on non-Windows systems where that path exists and is readable) instead of /dev/random as the primary source for secure random data. Attempts to read from /dev/random can block if the underlying system does not have sufficient entropy, which can have a severe adverse effect on performance. Reads from /dev/urandom will not block, and the data that it provides is no less secure than data from /dev/random in any way that matters for the server.
DS-42349, DS-43209, DS-43210, DS-43323, DS-43324	Added support for JSON-formatted audit loggers, which complement the existing file-based LDIF-formatted error logger. The JSON-formatted audit log messages provide a record of changes to data in the server and can be written to files on the local filesystem, written to the JVM's standard output or standard error stream, or sent to a syslog server. Added support for JSON-formatted HTTP operation loggers, which complement the existing file-based loggers using the W3C common log format and a proprietary space-delimited text format. The JSON-formatted HTTP operation log messages provide a record of interaction with HTTP clients and can be written to files on the local filesystem, written to the JVM's standard output or standard error stream, or sent to a syslog server. Fixed an issue that caused JSON-formatted loggers to use a timestamp format that was not strictly compliant with the ISO 8601 format described in RFC 3339. Timestamps incorrectly omitted the colon between the hour and minute components of the time zone offset.

Ticket ID	Description
DS-42381	Fixed an issue that could prevent the uninstaller from removing information about the instance from the topology registry.
DS-42447	Fixed an issue where PingDirectoryProxy Server sometimes logged a null pointer exception if search requests across multiple load-balanced backend servers resulted in a timeout.
DS-42504	Updated manage-profile replace-profile to set encryption settings definitions defined in the newer server profile as preferred in the encryption settings db.
DS-42527	Fixed an issue that could cause an exception when creating a resource in SCIM 1.1 using certain types of DNTemplate.
DS-42547	Fixed an issue where manage-profile generate-profile would print "null" as the generated profile directory when writing to an existing directory.
DS-42609	Fixed an issue in which the Directory REST API could fail to decode certain credentials when using basic authentication.
DS-42609	Fixed an issue in which the Consent API could fail to decode certain credentials when using basic authentication.
DS-42632	Added support for creating or importing a key pair configuration object using an elliptic curve (EC) key algorithm. You can use this to designate the encryption key pair for a JWT access token validator that handles EC-encrypted access tokens.
DS-42634	The JWT Access Token Validator can now validate JWT access tokens signed using the elliptic curve digital signature algorithms ES256, ES384, and ES512.
DS-42635	The JWT Access Token Validator can now validate JWT access tokens encrypted using elliptic curve cryptographic algorithms. The following key encryption algorithms are now supported in addition to RSA-OAEP: ECDH-ES, ECDH-ES+A128KW, ECDH-ES+A192KW, and ECDH-ES +A256KW. To support best practices for JWT security, you must now also configure the JWT Access Token Validator with explicit allow lists for key encryption and content encryption algorithms. For backward compatibility, the key encryption allow list defaults to RSA-OAEP, while the content encryption allow list defaults to A128CBC-HS256, A192CBC-HS384, and A256CBC-HS512. We recommend setting both allow lists to the strict minimum set of algorithms needed by the Access Token Validator.
DS-42651	Updated the manage-profile replace-profile subcommand to better support updating the server's keystore and truststore files. When using the --generateSelfSignedCertificate argument in a server profile's setup-arguments.txt file, the server will maintain the original keystore and truststore files during replace-profile. Otherwise, replace-profile will use the keystore and truststore specified in the profile's setup-arguments.txt file.
DS-42667	Updated the server to set a unique cluster name when started for the first time.
DS-42669, DS-42748	Updated the online dsconfig step of the manage-profile replace-profile subcommand to support getting LDAP connection arguments from a tools.properties file on the server being updated. Fixed an issue where boolean LDAP connection arguments like --useSSL and --trustAll would cause manage-profile replace-profile to fail when applying dsconfig online.

Ticket ID	Description
DS-42673	Updated the manage-profile setup subcommand to fail if the start-server command has a non-zero exit code.
DS-42681, DS-42684	Performance statistics generated by the Sideband API can now be published by the Periodic Stats Logger. To enable this, use the "included-http-servlet-stat" property of the Periodic Stats Logger.
DS-42687	Upgrade to Jetty 9.4.30
DS-42740	Fixed an issue where the **`dsconfig list`** subcommand would not display requested properties.
DS-42749	To support best practices for JWT security, you must now configure the JWT Access Token Validator with an explicit list of the JWT signing algorithms that it accepts. For backward compatibility, this list defaults to the RSA signing algorithms RS256, RS384, and RS512, but we recommend setting this list to the strict minimum set of signing algorithms needed by the Access Token Validator.
DS-42751	Added new override-status-code and additional-response-contents attributes to the Availability State HTTP Servlet Extension. These new attributes can be used to customize the response code and JSON response body of the servlet.
DS-42850	Fixed a typo in the password-expiring template that caused "password_expiration_time_of_day" to be printed instead of the password expiration time.
DS-42861	Updated the manage-profile tool logs to include the duration of each step the tool takes. The new --verbose argument can also be used to display timing information in the tool's console output.
DS-42872	Added a JSON-formatted stats logger to the server's default configuration. The stats logger is disabled by default.
DS-42886	Updated non-interactive setup (including manage-profile setup) to allow the password for the initial root user to be provided in pre-encoded form using the PBKDF2, SSHA256, SSHA384, or SSHA512 password storage scheme. This eliminates the need to have access to the clear-text password when setting up the server.
DS-42926	Fixed an issue where Ping Directory products configured to run as Microsoft Windows services were sometimes unable to automatically restart following an unplanned reboot, due to errors reading a corrupted server status file.
DS-42939	The Administrative Console configuration settings have been updated to account for the new SSO functionality.
DS-42952	For Windows only, there can be a hang on start when global configuration property `startup-error-logger-output-location` is set to values that contain `standard-error`. For Windows only, `standard-error` values are silently mapped to equivalent `standard-output` values.
DS-42963	Updated the manage-profile generate-profile subcommand to ignore files larger than 100 megabytes when generating a server profile. Fixed an issue where many large files in the server root could cause the tool to run out of memory.

Ticket ID	Description
DS-43027	Added a new --adminPasswordFile argument to the manage-topology add-server command, to allow specifying the administrator password with a file rather than with the command line.
DS-43073, DS-43198	Added support for ID Token Validators, which validate the integrity and content of ID tokens issued by OpenID Connect providers. Use these validators with the OAuth Bearer SASL Mechanism Handler to enable single sign-on (SSO) for the Administrative Console using an OpenID Connect provider such as PingOne. Currently, only PingOne is supported for SSO.
DS-43074	Added three built-in identity mappers that you can use to look up administrative accounts stored in the server configuration: Root DN Users, Topology Admin Users, and All Admin Users.
DS-43288	Updated setup and the replace-certificate tool to improve the way we generate self-signed certificates and certificate signing requests to make them more palatable to clients. To reduce the frequency with which administrators had to replace self-signed certificates, we previously used a very long lifetime for self-signed certificates generated by setup or the replace-certificate tool. However, some clients (especially web browsers and other HTTP clients) have started more strenuously objecting to certificates to long lifetimes, so we now generate self-signed certificates with a one-year validity period. The inter-server certificate (which is used internally within the server and does not get exposed to normal clients) is still created with a twenty-year lifetime. Also, the replace-certificate tool's interactive mode has been updated to improve the process that it uses to obtain information to include in the subject DN and subject alternative name extension for self-signed certificates and certificate signing requests. The following changes have been made in accordance with CA/Browser Forum guidelines: * When selecting the subject DN for the certificate, we listed a number of common attributes that may be used, including CN, OU, O, L, ST, and C. We previously indicated that CN attribute was recommended. We now also indicate that the O and C attributes are recommended as well. * When obtaining the list of DNS names to include in the subject alternative name extension, we previously suggested all names that we could find associated with interfaces on the local system. In many cases, we now omit non-qualified names and names that are associated with loopback interfaces. We will also warn about any attempts to add unqualified or invalid names to the list. * When obtaining the list of IP addresses to include in the subject alternative name extension, we previously suggested all addresses associated with all network interfaces on the system. We no longer suggest any IP addresses associated with loopback interfaces, and we no longer suggest any IP addresses associated in IANA-reserved ranges (for example, addresses reserved for private-use networks). The tool will now warn about attempts to add these addresses for inclusion in the subject alternative name extension.
DS-43305	Increased the maximum number of RDN components that a DN may have from 50 to 100.
DS-43376	Updated log publisher logic to reduce the amount of CPU that the server consumes when it is idle.
DS-43466	Updated PingDirectoryProxy so that in entry balanced environments, the JoinRequestControl on a search operation is passed through to the directory server instance in more scenarios. This enables more efficient processing, which leads to faster response times.

Ticket ID	Description
DS-43480	Updated the system information monitor provider to restrict the set of environment variables that may be included. Previously, the monitor entry included information about all defined environment variables, as that information can be useful for diagnostic purposes. However, some deployments may include credentials, secret keys, or other sensitive information in environment variables, and that should not be exposed in the monitor. The server will now only include values from a predefined set of environment variables that are expected to be the most useful for troubleshooting problems, and that are not expected to contain sensitive information.
DS-43517	Updated the jose4j library used for JWT signing and encryption to version 0.7.2.
DS-43651	The Security Guide is now available online at pingidentity.com. The guide has been removed from the server packaging.

Critical fixes

Critical Fixes

This release of the Directory Proxy Server addresses critical issues from earlier versions. Update all affected servers appropriately.

- Addressed an issue that could lead to slow, off-heap memory growth. This only occurred on servers whose cn=Version,cn=monitor entry was retrieved frequently.
  - Fixed in: 8.1.0.0
  - Introduced in: 5.2.0.0
  - Support identifiers: DS-41301
- Fixed a memory leak when performing SCIM queries on the Directory Server.
  - Fixed in: 8.1.0.0
  - Introduced in: 7.2.0.0
  - Support identifiers: DS-41206 SF#00681395
- The following enhancements were made to the topology manager to make it easier to diagnose connection errors:

  - Added monitoring information for all the failed outbound connections (including the time since it's been failing and the last error message seen when the failure occurred) from a server to one of its configured peers and the number of failed outbound connections.

  - Added alarms/alerts for when a server fails to connect to a peer server within a configured grace period.

  - Fixed in: 7.3.0.0
  - Introduced in: 7.0.0.0
  - Support identifiers: DS-38334 SF#00655578
- The topology manager will now raise a mirrored-subtree-manager-connection-asymmetry alarm when a server is able to establish outbound connections to its peer servers, but those peer servers are unable to establish connections back to the server within the configured grace period. The alarm is cleared as soon as there is connection symmetry.
  - Fixed in: 7.3.0.0
  - Introduced in: 7.0.0.0
  - Support identifiers: DS-38344 SF#00655578

- The dsreplication tool has been fixed to work when the node being used to enable replication is currently out-of-sync with the topology master.
  - Fixed in: 7.3.0.0
  - Introduced in: 7.0.0.0
  - Support identifiers: DS-38335 SF#00655578
- Fixed two issues in which the server could have exposed some clear-text passwords in files on the server file system.

  * When creating an encrypted backup of the alarms, alerts, configuration, encryption settings, schema, tasks, or trust store backends, the password used to generate the encryption key (which may have been obtained from an encryption settings definition) could have been inadvertently written into the backup descriptor. This problem does not affect local DB backends (like userRoot), the LDAP changelog backend, or the replication database.

  * When running certain command-line tools with an argument instructing the tool to read a password from a file, the password contained in that file could have been written into the server's tool invocation log instead of the path to that file. Affected tools include backup, create-initial-config, create-initial-proxy-config, dsreplication, enter-lockdown-mode, export-ldif, import-ldif, ldappasswordmodify, leave-lockdown-mode, manage-tasks, manage-topology, migrate-ldap-schema, parallel-update, prepare-endpoint-server, prepare-external-server, realtime-sync, rebuild-index, re-encode-entries, reload-http-connection-handler-certificates, reload-index, remove-defunct-server, restore, rotate-log, and stop-server. Other tools are not affected. Also note that this only includes passwords contained in files that were provided as command-line arguments; passwords included in the tools.properties file, or in a file referenced from tools.properties, would not have been exposed.

  In each of these cases, the files would have been written with permissions that make their contents only accessible to the system account used to run the server. Further, while administrative passwords may have been exposed in the tool invocation log, neither the passwords for regular users, nor any other data from their entries, should have been affected. We have introduced new automated tests to help ensure that such incidents do not occur in the future.

  We recommend changing any administrative passwords you fear may have been compromised as a result of this issue. If you are concerned that the passphrase for an encryption settings definition may have been exposed, then we recommend creating a new encryption settings definition that is preferred for all subsequent encryption operations, exporting your data to LDIF, and re-importing so that it will be encrypted with the new key. You also may wish to re-encrypt or destroy any existing backups, LDIF exports, or other data encrypted with a compromised key, and you may wish to sanitize or destroy any existing tool invocation log files that may contain clear-text passwords.
  - Fixed in: 7.3.0.0
  - Introduced in: 7.0.0.0
  - Support identifiers: DS-38897 DS-38908
- The following enhancements were made to the topology manager to make it easier to diagnose the connection errors:

  - Added monitoring information for all the failed outbound connections (including the time since it's been failing and the last error message seen when the failure occurred) from a server to one of its configured peers and the number of failed outbound connections.

  - Added alarms/alerts for when a server fails to connect to a peer server within a configured grace period.
  - Fixed in: 7.2.1.0
  - Introduced in: 7.0.0.0
  - Support identifiers: DS-38334 SF#00655578
- The topology manager will now raise a mirrored-subtree-manager-connection-asymmetry alarm when a server is able to establish outbound connections to its peer servers, but those peer servers are unable

to establish connections back to the server within the configured grace period. The alarm is cleared when connection symmetry is achieved.

  - Fixed in: 7.2.1.0
  - Introduced in: 7.0.0.0
  - Support identifiers: DS-38344 SF#00655578
- The dsreplication tool has been fixed to work when the node being used to enable replication is currently out-of-sync with the topology master.
  - Fixed in: 7.2.1.0
  - Introduced in: 7.0.0.0
  - Support identifiers: DS-38335 SF#00655578
- Fixed two issues in which the server could have exposed some clear-text passwords in files on the server file system.

  * When creating an encrypted backup of the alarms, alerts, configuration, encryption settings, schema, tasks, or trust store backends, the password used to generate the encryption key (which may have been obtained from an encryption settings definition) could have been inadvertently written into the backup descriptor. This problem does not affect local DB backends (like userRoot), the LDAP changelog backend, or the replication database.

  * When running certain command-line tools with an argument instructing the tool to read a password from a file, the password contained in that file could have been written into the server's tool invocation log instead of the path to that file. Affected tools include backup, create-initial-config, create-initial-proxy-config, dsreplication, enter-lockdown-mode, export-ldif, import-ldif, ldappasswordmodify, leave-lockdown-mode, manage-tasks, manage-topology, migrate-ldap-schema, parallel-update, prepare-endpoint-server, prepare-external-server, realtime-sync, rebuild-index, re-encode-entries, reload-http-connection-handler-certificates, reload-index, remove-defunct-server, restore, rotate-log, and stop-server. Other tools are not affected. Also note that this only includes passwords contained in files that were provided as command-line arguments; passwords included in the tools.properties file, or in a file referenced from tools.properties, would not have been exposed.

  In each of these cases, the files would have been written with permissions that make their contents only accessible to the system account used to run the server. Further, while administrative passwords may have been exposed in the tool invocation log, neither the passwords for regular users, nor any other data from their entries, should have been affected. We have introduced new automated tests to help ensure that such incidents do not occur in the future.

  We recommend changing any administrative passwords you fear may have been compromised as a result of this issue. If you are concerned that the passphrase for an encryption settings definition may have been exposed, then we recommend creating a new encryption settings definition that is preferred for all subsequent encryption operations, exporting your data to LDIF, and re-importing so that it will be encrypted with the new key. You also may wish to re-encrypt or destroy any existing backups, LDIF exports, or other data encrypted with a compromised key, and you may wish to sanitize or destroy any existing tool invocation log files that may contain clear-text passwords.

  - Fixed in: 7.0.1.3
  - Introduced in: 7.0.0.0
  - Support identifiers: DS-38897 DS-38908
- The server can now detect an "out of file handles" situation on the operating system, and shut down to prevent running in an unreliable state.
  - Fixed in: 5.1.0.0
  - Introduced in: 2.1.0.0
  - Support identifiers: DS-12579 SF#2655
- Disabled support for SSLv3 by default in the LDAP, HTTP, and JMX connection handlers, and for replication communication. The recently-discovered POODLE vulnerability could potentially allow a

network attacker to determine the plaintext behind an SSLv3-encrypted session, which would effectively negate the primary benefit of the encryption.

SSLv3 was initially defined in 1996, but was supplanted by the release of the TLSv1 definition in 1999 (and subsequently by TLSv1.1 in 2006 and TLSv1.2 in 2008). These newer TLS protocols are not susceptible to the POODLE vulnerability, and the server has supported them (and preferred them over SSLv3) for many years. The act of disabling SSLv3 by default should not have any adverse effect on clients that support any of the newer TLS protocols. However, if there are any legacy client applications that attempt to communicate securely but do not support the newer TLS protocols, they should be updated to support the newer protocols. In the event that there are known clients that do not support any security protocol newer than SSLv3 and that cannot be immediately updated to support a newer protocol, SSLv3 support can be re-enabled using the newly-introduced allowed-insecure-tls-protocol global configuration property. However, since communication using SSLv3 can no longer be considered secure, it is strongly recommended that every effort be made to update all known clients still using SSLv3.

It is possible to use the server access log to identify LDAP clients that use SSLv3 to communicate with the server. Whenever an LDAP client establishes a secure connection to the server, or whenever a client uses the StartTLS extended operation to secure an existing plaintext connection, the server will generate a SECURITY-NEGOTIATION access log message. The "protocol" element of a SECURITY-NEGOTIATION access log message specifies the name of the security protocol that has been negotiated between the client and the server, and any SECURITY-NEGOTIATION messages with a protocol of "SSLv3" suggest that the associated client is vulnerable to the POODLE attack. In addition, if any connections are terminated for attempting to use the disallowed SSLv3 protocol, the access log message for that disconnect should include a message stating the reason for the termination.

  - Fixed in: 5.0.0.0
  - Introduced in: 2.1.0.0
  - Support identifiers: DS-11782
- Update the replication backlog health check so that if a problem is encountered while attempting to retrieve monitor information from a backend server, that server will only be classified as degraded rather than unavailable.
  - Fixed in: 4.5.0.0
  - Introduced in: 4.1.0.7
  - Support identifiers: DS-9726
- Fix a bug that allows users with expired passwords to change attributes in their own entry other than password.
  - Fixed in: 3.5.0.0
  - Introduced in: 3.2.0.0
  - Support identifiers: DS-6054
- Update the PingDirectory Server to apply access controls when processing the GetAuthorizationEntryRequestControl.
  - Fixed in: 3.5.0.0
  - Introduced in: 2.0.0.0
  - Support identifiers: DS-854

Release Notes Archive

Release notes for previous versions of PingDirectoryProxy Server are included for reference.

PingDirectoryProxy Server 8.1.0.6 release notes
The release notes for the 8.1.0.6 release of PingDirectoryProxy Server.

Critical fixes

This release of the Directory Proxy Server addresses critical issues from earlier versions. Update all affected servers appropriately.

No critical issues have been identified.

PingDirectoryProxy Server 8.1.0.5 release notes

Critical fixes

This release of the Directory Proxy Server addresses critical issues from earlier versions. Update all affected servers appropriately.

No critical issues have been identified.

Resolved issues

The following issues have been resolved with this release of the Directory Proxy Server.

Ticket ID	Description
DS-43288	Updated setup and the replace-certificate tool to improve the way we generate self-signed certificates and certificate signing requests to make them more palatable to clients. To reduce the frequency with which administrators had to replace self-signed certificates, we previously used a very long lifetime for self-signed certificates generated by setup or the replace-certificate tool. However, some clients (especially web browsers and other HTTP clients) have started more strenuously objecting to certificates to long lifetimes, so we now generate self-signed certificates with a one-year validity period. The inter-server certificate (which is used internally within the server and does not get exposed to normal clients) is still created with a twenty-year lifetime. Also, the replace-certificate tool's interactive mode has been updated to improve the process that it uses to obtain information to include in the subject DN and subject alternative name extension for self-signed certificates and certificate signing requests. The following changes have been made in accordance with CA/Browser Forum guidelines: * When selecting the subject DN for the certificate, we listed a number of common attributes that may be used, including CN, OU, O, L, ST, and C. We previously indicated that CN attribute was recommended. We now also indicate that the O and C attributes are recommended as well. * When obtaining the list of DNS names to include in the subject alternative name extension, we previously suggested all names that we could find associated with interfaces on the local system. In many cases, we now omit non-qualified names and names that are associated with loopback interfaces. We will also warn about any attempts to add unqualified or invalid names to the list. * When obtaining the list of IP addresses to include in the subject alternative name extension, we previously suggested all addresses associated with all network interfaces on the system. We no longer suggest any IP addresses associated with loopback interfaces, and we no longer suggest any IP addresses associated in IANA-reserved ranges (for example, addresses reserved for private-use networks). The tool will now warn about attempts to add these addresses for inclusion in the subject alternative name extension.

Ticket ID	Description
DS-44204	Fixed an issue in which the Directory Server could incorrectly allow requests to be processed with an alternate authorization identity (for example, using the proxied authorization control, or if the requests pass through a Directory Proxy Server) whose account is in a "must change password" state. The server will now only permit the operation if it attempts to set a new password for the target user.
DS-44316	Reduced the JVM memory requirements for many command line tools. This avoids memory pressure when multiple tools, such as a scheduled collect-support-data task, are run concurrently to the server process. For most tools, the initial heap size has been reduced to 128 MB, and for certain tools the maximum heap size has capped at 512 MB. On systems with larger amounts of memory, these tools previously were allotted unnecessarily large heaps. The maximum heap size has not been reduced for any tool that especially benefits from having more memory.

PingDirectoryProxy Server 8.1.0.2 Release Notes

Critical Fixes

This release of PingDirectoryProxy Server addresses critical issues from earlier versions. Update all affected servers appropriately.

- Addressed an issue that could lead to slow, off-heap memory growth. This only occurred on servers whose cn=Version,cn=monitor entry was retrieved frequently.
  - Fixed in: 8.1.0.0
  - Introduced in: 5.2.0.0
  - Support identifiers: DS-41301
- Fixed a memory leak when performing SCIM queries on the directory server.
  - Fixed in: 8.1.0.0
  - Introduced in: 7.2.0.0
  - Support identifiers: DS-41206 SF#00681395
- The following enhancements were made to the topology manager to make it easier to diagnose connection errors:
  - Added monitoring information for all the failed outbound connections (including the time since it has been failing and the last error message seen when the failure occurred) from a server to one of its configured peers and the number of failed outbound connections.
  - Added alarms/alerts for when a server fails to connect to a peer server within a configured grace period.
  - Fixed in: 7.3.0.0
  - Introduced in: 7.0.0.0
  - Support identifiers: DS-38334 SF#00655578
- The topology manager will now raise a mirrored-subtree-manager-connection-asymmetry alarm when a server is able to establish outbound connections to its peer servers, but those peer servers are unable

to establish connections back to the server within the configured grace period. The alarm is cleared as soon as there is connection symmetry.

  - Fixed in: 7.3.0.0
  - Introduced in: 7.0.0.0
  - Support identifiers: DS-38344 SF#00655578
- The dsreplication tool has been fixed to work when the node being used to enable replication is currently out-of-sync with the topology master.
  - Fixed in: 7.3.0.0
  - Introduced in: 7.0.0.0
  - Support identifiers: DS-38335 SF#00655578
- Fixed two issues in which the server could have exposed some clear-text passwords in files on the server file system.
  - Fixed in: 7.3.0.0
  - Introduced in: 7.0.0.0
  - Support identifiers: DS-38897 DS-38908
- The following enhancements were made to the topology manager to make it easier to diagnose the connection errors:
  - Fixed in: 7.2.1.0
  - Introduced in: 7.0.0.0
  - Support identifiers: DS-38334 SF#00655578
- The topology manager will now raise a mirrored-subtree-manager-connection-asymmetry alarm when a server is able to establish outbound connections to its peer servers, but those peer servers are unable to establish connections back to the server within the configured grace period. The alarm is cleared when connection symmetry is achieved.
  - Fixed in: 7.2.1.0
  - Introduced in: 7.0.0.0
  - Support identifiers: DS-38344 SF#00655578
- The dsreplication tool has been fixed to work when the node being used to enable replication is currently out-of-sync with the topology master.
  - Fixed in: 7.2.1.0
  - Introduced in: 7.0.0.0
  - Support identifiers: DS-38335 SF#00655578
- Fixed two issues in which the server could have exposed some clear-text passwords in files on the server file system.
  - Fixed in: 7.0.1.3
  - Introduced in: 7.0.0.0
  - Support identifiers: DS-38897 DS-38908
- The server can now detect an "out of file handles" situation on the operating system, and shut down to prevent running in an unreliable state.
  - Fixed in: 5.1.0.0
  - Introduced in: 2.1.0.0
  - Support identifiers: DS-12579 SF#2655
- Disabled support for SSLv3 by default in the LDAP, HTTP, and JMX connection handlers, and for replication communication. The recently-discovered POODLE vulnerability could potentially allow a

network attacker to determine the plaintext behind an SSLv3-encrypted session, which would effectively negate the primary benefit of the encryption.

  - Fixed in: 5.0.0.0
  - Introduced in: 2.1.0.0
  - Support identifiers: DS-11782
- Update the replication backlog health check so that if a problem is encountered while attempting to retrieve monitor information from a backend server, that server will only be classified as degraded rather than unavailable.
  - Fixed in: 4.5.0.0
  - Introduced in: 4.1.0.7
  - Support identifiers: DS-9726
- Fix a bug that allows users with expired passwords to change attributes in their own entry other than password.
  - Fixed in: 3.5.0.0
  - Introduced in: 3.2.0.0
  - Support identifiers: DS-6054
- Update PingDirectory Server to apply access controls when processing the GetAuthorizationEntryRequestControl.
  - Fixed in: 3.5.0.0
  - Introduced in: 2.0.0.0
  - Support identifiers: DS-854

Resolved Issues

The following issues have been resolved with this release of PingDirectoryProxy Server:

Ticket ID	Description
DS-40828	Fixed an issue where some state associated with a JMX connection was not freed after the connection was closed. This led to a slow memory leak in servers that were monitored by an application that created a new JMX connection each polling interval.
DS-41964	Fixed an issue with the manage-profile tool where files in a server profile's dsconfig/ directory without a ".dsconfig" extension could cause failures in manage-profile replace-profile when validating updated dsconfig files.
DS-42687	Upgrade to Jetty 9.4.30

PingDirectoryProxy Server 8.1 Release Notes

Critical Fixes

This release of PingDirectoryProxy Server addresses critical issues from earlier versions. Update all affected servers appropriately.

- Addressed an issue that could lead to slow, off-heap memory growth. This only occurred on servers whose cn=Version,cn=monitor entry was retrieved frequently.
  - Fixed in: 8.1.0.0
  - Introduced in: 5.2.0.0
  - Support identifiers: DS-41301
- Fixed a memory leak when performing SCIM queries on the Directory Server.
  - Fixed in: 8.1.0.0
  - Introduced in: 7.2.0.0
  - Support identifiers: DS-41206 SF#00681395
- Fixed a memory leak when performing SCIM queries on the Directory Server.
  - Fixed in: 8.0.0.1

    Introduced in: 7.2.0.0

    Support identifiers: DS-41206 SF#00681395
- Addressed an issue that could lead to slow off-heap memory growth. This only occurred on servers whose cn=Version,cn=monitor entry was retrieved frequently.
  - Fixed in: 8.0.0.1

    Introduced in: 5.2.0.0

    Support identifiers: DS-41301
- The following enhancements were made to the topology manager to make it easier to diagnose connection errors:

  - Added monitoring information for all the failed outbound connections (including the time since it's been failing and the last error message seen when the failure occurred) from a server to one of its configured peers and the number of failed outbound connections.

  - Added alarms/alerts for when a server fails to connect to a peer server within a configured grace period.
  - Fixed in: 7.3.0.0
  - Introduced in: 7.0.0.0
  - Support identifiers: DS-38334 SF#00655578
- The topology manager will now raise a mirrored-subtree-manager-connection-asymmetry alarm when a server is able to establish outbound connections to its peer servers, but those peer servers are unable to establish connections back to the server within the configured grace period. The alarm is cleared as soon as there is connection symmetry.
  - Fixed in: 7.3.0.0
  - Introduced in: 7.0.0.0
  - Support identifiers: DS-38344 SF#00655578
- The dsreplication tool has been fixed to work when the node being used to enable replication is currently out-of-sync with the topology master.
  - Fixed in: 7.3.0.0
  - Introduced in: 7.0.0.0
  - Support identifiers: DS-38335 SF#00655578
- Fixed two issues in which the server could have exposed some clear-text passwords in files on the server file system.

  * When creating an encrypted backup of the alarms, alerts, configuration, encryption settings, schema, tasks, or trust store backends, the password used to generate the encryption key (which may have been obtained from an encryption settings definition) could have been inadvertently written into

the backup descriptor. This problem does not affect local DB backends (like userRoot), the LDAP changelog backend, or the replication database.

* When running certain command-line tools with an argument instructing the tool to read a password from a file, the password contained in that file could have been written into the server's tool invocation log instead of the path to that file. Affected tools include backup, create-initial-config, create-initial-proxy-config, dsreplication, enter-lockdown-mode, export-ldif, import-ldif, ldappasswordmodify, leave-lockdown-mode, manage-tasks, manage-topology, migrate-ldap-schema, parallel-update, prepare-endpoint-server, prepare-external-server, realtime-sync, rebuild-index, re-encode-entries, reload-http-connection-handler-certificates, reload-index, remove-defunct-server, restore, rotate-log, and stop-server. Other tools are not affected. Also note that this only includes passwords contained in files that were provided as command-line arguments; passwords included in the tools.properties file, or in a file referenced from tools.properties, would not have been exposed.

In each of these cases, the files would have been written with permissions that make their contents only accessible to the system account used to run the server. Further, while administrative passwords may have been exposed in the tool invocation log, neither the passwords for regular users, nor any other data from their entries, should have been affected. We have introduced new automated tests to help ensure that such incidents do not occur in the future.

We recommend changing any administrative passwords you fear may have been compromised as a result of this issue. If you are concerned that the passphrase for an encryption settings definition may have been exposed, then we recommend creating a new encryption settings definition that is preferred for all subsequent encryption operations, exporting your data to LDIF, and re-importing so that it will be encrypted with the new key. You also may wish to re-encrypt or destroy any existing backups, LDIF exports, or other data encrypted with a compromised key, and you may wish to sanitize or destroy any existing tool invocation log files that may contain clear-text passwords.

  - Fixed in: 7.3.0.0
  - Introduced in: 7.0.0.0
  - Support identifiers: DS-38897 DS-38908
- The following enhancements were made to the topology manager to make it easier to diagnose the connection errors:

  - Added monitoring information for all the failed outbound connections (including the time since it's been failing and the last error message seen when the failure occurred) from a server to one of its configured peers and the number of failed outbound connections.

  - Added alarms/alerts for when a server fails to connect to a peer server within a configured grace period.

  - Fixed in: 7.2.1.0
  - Introduced in: 7.0.0.0
  - Support identifiers: DS-38334 SF#00655578
- The topology manager will now raise a mirrored-subtree-manager-connection-asymmetry alarm when a server is able to establish outbound connections to its peer servers, but those peer servers are unable to establish connections back to the server within the configured grace period. The alarm is cleared when connection symmetry is achieved.

  - Fixed in: 7.2.1.0
  - Introduced in: 7.0.0.0
  - Support identifiers: DS-38344 SF#00655578
- The dsreplication tool has been fixed to work when the node being used to enable replication is currently out-of-sync with the topology master.

  - Fixed in: 7.2.1.0
  - Introduced in: 7.0.0.0
  - Support identifiers: DS-38335 SF#00655578

- Fixed two issues in which the server could have exposed some clear-text passwords in files on the server file system.

  * When creating an encrypted backup of the alarms, alerts, configuration, encryption settings, schema, tasks, or trust store backends, the password used to generate the encryption key (which may have been obtained from an encryption settings definition) could have been inadvertently written into the backup descriptor. This problem does not affect local DB backends (like userRoot), the LDAP changelog backend, or the replication database.

  * When running certain command-line tools with an argument instructing the tool to read a password from a file, the password contained in that file could have been written into the server's tool invocation log instead of the path to that file. Affected tools include backup, create-initial-config, create-initial-proxy-config, dsreplication, enter-lockdown-mode, export-ldif, import-ldif, ldappasswordmodify, leave-lockdown-mode, manage-tasks, manage-topology, migrate-ldap-schema, parallel-update, prepare-endpoint-server, prepare-external-server, realtime-sync, rebuild-index, re-encode-entries, reload-http-connection-handler-certificates, reload-index, remove-defunct-server, restore, rotate-log, and stop-server. Other tools are not affected. Also note that this only includes passwords contained in files that were provided as command-line arguments; passwords included in the tools.properties file, or in a file referenced from tools.properties, would not have been exposed.

  In each of these cases, the files would have been written with permissions that make their contents only accessible to the system account used to run the server. Further, while administrative passwords may have been exposed in the tool invocation log, neither the passwords for regular users, nor any other data from their entries, should have been affected. We have introduced new automated tests to help ensure that such incidents do not occur in the future.

  We recommend changing any administrative passwords you fear may have been compromised as a result of this issue. If you are concerned that the passphrase for an encryption settings definition may have been exposed, then we recommend creating a new encryption settings definition that is preferred for all subsequent encryption operations, exporting your data to LDIF, and re-importing so that it will be encrypted with the new key. You also may wish to re-encrypt or destroy any existing backups, LDIF exports, or other data encrypted with a compromised key, and you may wish to sanitize or destroy any existing tool invocation log files that may contain clear-text passwords.

  - Fixed in: 7.0.1.3
  - Introduced in: 7.0.0.0
  - Support identifiers: DS-38897 DS-38908

Upgrade Considerations

Important considerations for upgrading to this version of PingDirectoryProxy Server:

- If you have upgraded a server that is in a cluster (i.e., has a cluster name set in the Server Instance configuration object) to version 8.1, you will not be able to make cluster configuration changes until all servers with the same cluster name have been upgraded to version 8.1. If needed, you could create temporary clusters based on server versions and modify each of the servers' cluster name appropriately to minimize the impact while you are upgrading.
- Updated the "delay bind response" failure lockout action to provide an option to delay the response to bind requests initiated by non-LDAP clients (for example, when using HTTP basic authentication). This option is disabled by default because delaying the bind response for non-LDAP clients may require temporarily blocking the thread used to process the request, which could increase the risk of a denial-of-service attack. To help mitigate this risk, if you enable delayed bind responses for non LDAP clients, we recommend that you also increase the number of request handler threads for all enabled HTTP connection handlers.
- Updated setup to create a second encryption settings definition if data encryption is enabled. It will continue to create a definition for 128-bit AES encryption for use as the preferred definition to preserve backward compatibility with existing servers in the topology, but it will now also generate a definition for 256-bit AES encryption. The 256-bit AES definition may become the preferred definition in a future release, but you can use it now by first ensuring that any existing instances are updated to contain the

new definition (with the "encryption-settings export" and "encryption-settings import" commands) and then making it the preferred definition (with "encryption-settings set-preferred") in all instances.
- Fixed an issue that could prevent the uninstaller from removing information about the instance from the topology registry.

What's New

These are new features for this release of PingDirectoryProxy Server:

- The SCIM v2 REST API was added to PingDirectory Server in version 8.0. PingDirectoryProxy Server now supports the SCIMv2 REST API to create, read, update and delete (CRUD) users and other resources using JSON over HTTP. PingDirectoryProxy Server supports all the same endpoints and HTTP methods that PingDirectory Server supports.
- In an ongoing effort to improve the use of containers for PingDirectory Server, several features have been implemented:

  - The --outputFile option has been added to the collect-support-data tool. You can now specify either a path, a file name, or a path and file name for the resulting CSD file. This means an administrator can run the collect-support-data tool and put the output file into a directory outside of the container, allowing access to the file without having to actually connect to the container.

  - The collect-support-data tool can now be run as a recurring task. Recurring tasks can be created using the Administration console which means that administrators do not have to connect to the container in order to run the tool.

  - A Collect Support Tool Extended Operation has been added allowing LDAP clients to initiate the collect-support-data tool and to receive the output of the request. The LDAP SDK has been updated to support this, and the --remoteServer added to the collect-support-data tool can be used to send the request to another server. In other words, you can now run collect-support-data on the command line and reference another server, possibly in a container, and retrieve the output file remotely.
- PingDirectoryProxy Server has a Consent REST API that allows users to create and store consents. A new feature now allows users to search for consents that have been granted to them by another party.
- Updated PingDirectoryProxy Server to improve support for the PLAIN, UNBOUNDID-DELIVERED-OTP, UNBOUNDID-TOTP, and UNBOUNDID-YUBIKEY-OTP SASL mechanisms. Previously, PingDirectoryProxy Server itself performed all of the processing for those SASL mechanisms, and it would only work if PingDirectoryProxy Server could retrieve the appropriate encoded credentials from the backend PingDirectory Server. It will now forward the bind request to the backend server for processing, which allows it to work in deployments in which the backend server prevents PingDirectoryProxy Server from accessing the stored credentials.
- Added a new validate-ldap-schema tool that can be used to examine schema definitions in a set of LDIF files and report any issues that it detects.

Known Issues/Workarounds

The following are known issues in the current version of the PingDirectoryProxy Server:

- Several known issues can occur when you use the Administrative Console with Tomcat 9.0.31. You can resolve these issues by upgrading to Tomcat 9.0.33 or later.
- If you use the create-systemd-script tool to create a forking systemd service, the service is stopped by the "systemctl stop ping-directory.service" command. At that time, you can see the status using the "systemctl status ping-directory.service" command. That status might contain an indication of failure: "Active: failed (Result: exit-code)". This error has to do with the way the service exits. It is harmless.

Resolved Issues

The following issues have been resolved with this release of the PingDirectoryProxy Server:

Ticket ID	Description
DS-1046,DS-1204,DS-36547	Added support for remotely invoking the collect-support-data tool using an administrative task, and for invoking the tool on a regular basis as a recurring task. The tool has also been updated to add an outputPath argument to allow specifying the path or name to use for the output file.
DS-10216	Updated the GSSAPI SASL mechanism handler to support integrity and confidentiality protection for client communication.
DS-37829	The "create-systemd-script" CLI now creates a "forking" service file since Ping services are started by a process (the "start-server" script) that is different than the actual service process.
DS-38008,DS-41192	Updated PingDirectoryProxy Server to improve its support for retrying operations that fail in a manner that suggests they may succeed if attempted in a different server or on a newly established connection. Previously, PingDirectoryProxy Server would automatically enable retry support for all types of operations except add. At the time this retry support was initially implemented, retrying an add operation on a different server had the potential to result in a replication conflict if the entry was ultimately created on multiple servers. The Directory Server's replication logic has since been updated to detect and automatically resolve conflicts that result from attempts to add the same entry concurrently on multiple servers. As it is safe to enable automatic retry support for add operations (along with all other types of operations), that is now the default configuration. PingDirectoryProxy Server has also been updated to improve its retry support for the case in which connections to multiple backend servers have become invalidated in a manner that it cannot immediately detect (for example, as the result of an issue with a network switch or hardware load-balancer). As a result of this change, if all of the PingDirectoryProxy Server's usual attempts to process an operation fail in a manner that could indicate that connections are no longer valid, it may now attempt to establish a new connection to a backend server for one additional retry attempt.

Ticket ID	Description
DS-38122	Added support for an extended operation that can be used to invoke the collect-support-data tool from a remote system and stream the output and resulting support data archive back to the client. The collect-support-data command-line tool has been updated to support this capability through the new --useRemoteServer argument.
DS-38535	Fixed an issue that could cause the server to generate an administrative alert about an uncaught exception when trying to send data on a TLS-encrypted connection that is no longer valid.
DS-39798	Fixed a bug in which SEMI_AGGRESSIVE and AGGRESSIVE JVM Tuning Parameters were previously allowed to both be selected.
DS-40356	Updated the manage-profile tool to prevent displaying warnings about offline config changes when starting the server.
DS-40532	Added a logging-error-behavior property to the log publisher, periodic stats logger plugin, and monitor history plugin configuration that can be used to specify the behavior the server should exhibit if an error occurs while attempting logging-related processing. By default, the server will preserve its previous behavior of writing a message to standard error, but it can be configured to enter lockdown mode on a logging error, in which the server will report itself as unavailable and will only accept requests from accounts with the lockdown-mode privilege and only from clients communicating over a loopback interface.
DS-40551	Fixed an issue that could prevent some tools from running properly with an encrypted tools.properties file.
DS-40567	A license is now always required when using the manage-profile replace-profile tool.
DS-40681	Added a cache for password policies stored in user data rather than in the configuration. The cache will hold up to 500 policies by default, but the cache size can be configured (or the cache disabled) using the maximum-user-data-password-policies-to-cache property in the global configuration.
DS-40746	Updated the logic that the server uses to select an appropriate default set of TLS cipher suites.

Ticket ID	Description
DS-40806	Fixed an issue that could cause the shutdown process to stall if the server is configured to use TCP to communicate with a StatsD endpoint that has become unresponsive.
DS-40817	PATCH operations on SCIM 2 for PingDirectory Server now require that the value of the schemas attribute in the request body to be "urn:ietf:params:scim:api:messages:2.0:PatchOp", in accordance with RFC 7644.
DS-40889	Fixed an issue with recurring exec tasks where the working-directory attribute was ignored.
DS-40953	Added detection for buffer issues that could cause connections to get stuck during TLS handshake.
DS-41054	Fixed an issue that stopped new extensions from being installed.
DS-41074	Fixed an issue with the way the server reports memory usage after completing an explicitly requested garbage collection.
DS-41086	Updated the StatsD monitoring endpoint to replace any spaces, commas, or colons with underscores, and remove and single quotes or double quotes in sent metric lines. This simplifies parsing of the produced metrics.
DS-41118	A gauge called HTTP Processing (Percent) is now available. This gauge measures the server's capacity to process new incoming HTTP requests.
DS-41126	Updated the server to make the general monitor entry available to JMX clients.
DS-41136	Enabled SCIM 2 API for PingDirectoryProxy Server.
DS-41142	Improved debugging support for Server SDK extensions. If debugging is enabled, the server will now generate a debug message whenever it invokes an extension. For some extension methods that return a value, the server will also generate a debug message with that return value.
DS-41206	Fixed a memory leak when performing SCIM queries on PingDirectory Server.

Ticket ID	Description
DS-41235	Updated the cn=Cluster subtree to prevent clustered configuration changes when servers in the cluster have mixed versions. To make clustered configuration changes, either update all servers in the cluster to the same version, or temporarily create separate clusters by server version by changing the cluster-name property on the server instance configuration objects.
DS-41236	To avoid inconsistencies, changing clustered configuration will now require all servers in the cluster to be on the same product version. Servers will not pull any clustered configuration from the master of the cluster if they are on a different product version.
DS-41261	Fixed an issue with manage-profile replace-profile where certain configuration changes for recurring task chains were not being applied.
DS-41289	Fixed an issue that prevented password changes for topology administrators unless their password policy was configured to allow pre-encoded passwords.
DS-41299	Fixed an issue where the Ping Directory Proxy could incorrectly return duplicate entries after timing out on an unindexed search.
DS-41301	Addressed an issue that could lead to slow, off-heap memory growth. This only occurred on servers whose cn=Version,cn=monitor entry was retrieved frequently.
DS-41333	Added an ssl-client-auth-policy configuration property to the HTTP connection handler to provide support for mutual TLS authentication.
DS-41366	Updated the base monitor entry to include locationName and locationDN attributes if the server is configured with a location.
DS-41396	Updated the Server SDK to add ClientContext and OperationContext methods for obtaining the name and DN of the associated client connection policy.
DS-41400	Updated the file servlet HTTP servlet extension to add support for requiring authentication in order to access the content. Access may optionally be limited to members of a specified set of groups.
DS-41622	The use-administrative-operation-request-control property is now hidden on unsupported products.

Ticket ID	Description
DS-41731	Fixed an issue that could prevent setup from generating a self-signed certificate for systems with non-ASCII hostnames.
DS-41762	Fixed an issue where mirrored subtree polling could produce config archive files that were identical or ignored the configured insignificant attributes list.
DS-41818	Added the --zip argument to the manage-profile generate-profile subcommand, which can be used to generate a zipped server profile.
DS-41820	Added an administrative task that may be used to generate a server profile and a corresponding recurring task that may be used to invoke the task on a regular basis.
DS-41821	Added an instance root file servlet to the default configuration. HTTPS requests to /instance-root by authenticated users with the file-servlet-access privilege will be granted access to files within the server instance root.
DS-41850	Servers running on Linux will now log a warning about possible performance impacts if the current memory control group has memory.swappiness set to a nonzero value.
DS-41851	Enabled Correlated LDAP Data Views for SCIM 2 resource types on PingDirectory Server and Nokia8661DirectoryProxyServerAdministrationGuid.
DS-41987	Updated the PUT request in the consent service to reject requests that have duplicate collaborators to make it consistent with POST and PATCH requests.
DS-42006	The server now warns the administrator at startup if there are multiple versions of the same jar listed in the classpath, and the first one in the classpath is not the newest one.
DS-42033	Addressed an issue where some tools would throw a NullPointerException if a server was configured with a custom global result code map.
DS-42387	Updated the **`manage-profile generate-profile`** subcommand to exclude files in the `ldif/` and `bak/` directories by default when generating a server profile. If necessary, you can manually include those directories using the `--includePath` argument.

PingDirectoryProxy Server 8.0.0.5 release notes

Critical fixes

This release of the Directory Proxy Server addresses critical issues from earlier versions. Update all affected servers appropriately.

No critical issues have been identified.

Resolved issues

The following issues have been resolved with this release of the Directory Proxy Server.

Ticket ID	Description
DS-44204	Fixed an issue in which the Directory Server could incorrectly allow requests to be processed with an alternate authorization identity (for example, using the proxied authorization control, or if the requests pass through a Directory Proxy Server) whose account is in a "must change password" state. The server will now only permit the operation if it attempts to set a new password for the target user.

PingDirectoryProxy Server 8.0.0.3 release notes

Critical Fixes

This release of the PingDirectoryProxy Server addresses critical issues from earlier versions. Update all affected servers appropriately.

No critical issues have been identified

Resolved Issues

The following issues have been resolved with this release of the PingDirectoryProxy Server:

Ticket ID	Description
DS-38535	Fixed an issue that could cause the server to generate an administrative alert about an uncaught exception when trying to send data on a TLS-encrypted connection that is no longer valid.

Ticket ID	Description
DS-43288	Updated setup and the replace-certificate tool to improve the way we generate self-signed certificates and certificate signing requests to make them more palatable to clients. To reduce the frequency with which administrators had to replace self-signed certificates, we previously used a very long lifetime for self-signed certificates generated by setup or the replace-certificate tool. However, some clients (especially web browsers and other HTTP clients) have started more strenuously objecting to certificates to long lifetimes, so we now generate self-signed certificates with a one-year validity period. The inter-server certificate (which is used internally within the server and does not get exposed to normal clients) is still created with a twenty-year lifetime. Also, the replace-certificate tool's interactive mode has been updated to improve the process that it uses to obtain information to include in the subject DN and subject alternative name extension for self-signed certificates and certificate signing requests. The following changes have been made in accordance with CA/Browser Forum guidelines: * When selecting the subject DN for the certificate, we listed a number of common attributes that may be used, including CN, OU, O, L, ST, and C. We previously indicated that CN attribute was recommended. We now also indicate that the O and C attributes are recommended as well. * When obtaining the list of DNS names to include in the subject alternative name extension, we previously suggested all names that we could find associated with interfaces on the local system. In many cases, we now omit non-qualified names and names that are associated with loopback interfaces. We will also warn about any attempts to add unqualified or invalid names to the list. * When obtaining the list of IP addresses to include in the subject alternative name extension, we previously suggested all addresses associated with all network interfaces on the system. We no longer suggest any IP addresses associated with loopback interfaces, and we no longer suggest any IP addresses associated in IANA-reserved ranges (for example, addresses reserved for private-use networks). The tool will now warn about attempts to add these addresses for inclusion in the subject alternative name extension.

Ticket ID	Description
DS-43480	Updated the system information monitor provider to restrict the set of environment variables that may be included. Previously, the monitor entry included information about all defined environment variables, as that information can be useful for diagnostic purposes. However, some deployments may include credentials, secret keys, or other sensitive information in environment variables, and that should not be exposed in the monitor. The server will now only include values from a predefined set of environment variables that are expected to be the most useful for troubleshooting problems, and that are not expected to contain sensitive information.
DS-43632	Fixed an issue where the "format" field is omitted from the list of operational attribute schemas in the Directory REST API.
DS-43651	The Security Guide is now available online at pingidentity.com and has been removed from the server packaging.

PingDirectoryProxy Server 8.0.0.2 Release Notes

Critical Fixes

This release of PingDirectoryProxy Server addresses critical issues from earlier versions. Update all affected servers appropriately.

No critical issues have been identified

Resolved Issues

The following issues have been resolved with this release of PingDirectoryProxy Server:

Ticket ID	Description
DS-40551	Fixed an issue that could prevent some tools from running properly with an encrypted tools.properties file.
DS-40828	Fixed an issue where some state associated with a JMX connection was not freed after the connection was closed. This led to a slow memory leak in servers that were monitored by an application that created a new JMX connection each polling interval.
DS-41054	Fixed an issue that stopped new extensions from being installed.

Ticket ID	Description
DS-41074	Fixed an issue with the way the server reports memory usage after completing an explicitly requested garbage collection.
DS-41126	Updated the server to make the general monitor entry available to JMX clients.
DS-41235	Updated the cn=Cluster subtree to prevent clustered configuration changes when servers in the cluster have mixed versions. To make clustered configuration changes, either update all servers in the cluster to the same version, or temporarily create separate clusters by server version by changing the cluster-name property on the server instance configuration objects.
DS-41236	To avoid inconsistencies, changing clustered configuration will now require all servers in the cluster to be on the same product version. Servers will not pull any clustered configuration from the master of the cluster if they are on a different product version.
DS-41261	Fixed an issue with manage-profile replace-profile where certain configuration changes for recurring task chains were not being applied.
DS-41289	Fixed an issue that prevented password changes for topology administrators unless their password policy was configured to allow pre-encoded passwords.
DS-41299	Fixed an issue where PingDirectoryProxy Server could incorrectly return duplicate entries after timing out on an unindexed search.
DS-42609	Fixed an issue in which the Directory REST API could fail to decode certain credentials when using basic authentication.
DS-42812	Upgrade to jetty 9.4.30

PingDirectoryProxy Server 8.0.0.1 Release Notes

Critical Fixes

This release of the PingDirectoryProxy Server addresses critical issues from earlier versions. Update all affected servers appropriately.

- Fixed a memory leak when performing SCIM queries on the Directory Server.
  - Fixed in: 8.0.0.1

    Introduced in: 7.2.0.0

    Support identifiers: DS-41206 SF#00681395
- Addressed an issue that could lead to slow off-heap memory growth. This only occurred on servers whose cn=Version,cn=monitor entry was retrieved frequently.
  - Fixed in: 8.0.0.1

    Introduced in: 5.2.0.0

    Support identifiers: DS-41301
- The following enhancements were made to the topology manager to make it easier to diagnose connection errors:

  - Added monitoring information for all the failed outbound connections (including the time since it's been failing and the last error message seen when the failure occurred) from a server to one of its configured peers and the number of failed outbound connections.

  - Added alarms/alerts for when a server fails to connect to a peer server within a configured grace period.

  - Fixed in: 7.3.0.0
  - Introduced in: 7.0.0.0
  - Support identifiers: DS-38334 SF#00655578
- The topology manager will now raise a mirrored-subtree-manager-connection-asymmetry alarm when a server is able to establish outbound connections to its peer servers, but those peer servers are unable to establish connections back to the server within the configured grace period. The alarm is cleared as soon as there is connection symmetry.
  - Fixed in: 7.3.0.0
  - Introduced in: 7.0.0.0
  - Support identifiers: DS-38344 SF#00655578
- The dsreplication tool has been fixed to work when the node being used to enable replication is currently out-of-sync with the topology master.
  - Fixed in: 7.3.0.0
  - Introduced in: 7.0.0.0
  - Support identifiers: DS-38335 SF#00655578
- Fixed two issues in which the server could have exposed some clear-text passwords in files on the server file system.

  * When creating an encrypted backup of the alarms, alerts, configuration, encryption settings, schema, tasks, or trust store backends, the password used to generate the encryption key (which may have been obtained from an encryption settings definition) could have been inadvertently written into the backup descriptor. This problem does not affect local DB backends (like userRoot), the LDAP changelog backend, or the replication database.

  * When running certain command-line tools with an argument instructing the tool to read a password from a file, the password contained in that file could have been written into the server's tool invocation log instead of the path to that file. Affected tools include backup, create-initial-config, create-initial-proxy-config, dsreplication, enter-lockdown-mode, export-ldif, import-ldif, ldappasswordmodify, leave-

lockdown-mode, manage-tasks, manage-topology, migrate-ldap-schema, parallel-update, prepare-endpoint-server, prepare-external-server, realtime-sync, rebuild-index, re-encode-entries, reload-http-connection-handler-certificates, reload-index, remove-defunct-server, restore, rotate-log, and stop-server. Other tools are not affected. Also note that this only includes passwords contained in files that were provided as command-line arguments; passwords included in the tools.properties file, or in a file referenced from tools.properties, would not have been exposed.

In each of these cases, the files would have been written with permissions that make their contents only accessible to the system account used to run the server. Further, while administrative passwords may have been exposed in the tool invocation log, neither the passwords for regular users, nor any other data from their entries, should have been affected. We have introduced new automated tests to help ensure that such incidents do not occur in the future.

We recommend changing any administrative passwords you fear may have been compromised as a result of this issue. If you are concerned that the passphrase for an encryption settings definition may have been exposed, then we recommend creating a new encryption settings definition that is preferred for all subsequent encryption operations, exporting your data to LDIF, and re-importing so that it will be encrypted with the new key. You also may wish to re-encrypt or destroy any existing backups, LDIF exports, or other data encrypted with a compromised key, and you may wish to sanitize or destroy any existing tool invocation log files that may contain clear-text passwords.

- Fixed in: 7.3.0.0
- Introduced in: 7.0.0.0
- Support identifiers: DS-38897 DS-38908

- The following enhancements were made to the topology manager to make it easier to diagnose the connection errors:

  - Added monitoring information for all the failed outbound connections (including the time since it's been failing and the last error message seen when the failure occurred) from a server to one of its configured peers and the number of failed outbound connections.

  - Added alarms/alerts for when a server fails to connect to a peer server within a configured grace period.

  - Fixed in: 7.2.1.0
  - Introduced in: 7.0.0.0
  - Support identifiers: DS-38334 SF#00655578
- The topology manager will now raise a mirrored-subtree-manager-connection-asymmetry alarm when a server is able to establish outbound connections to its peer servers, but those peer servers are unable to establish connections back to the server within the configured grace period. The alarm is cleared when connection symmetry is achieved.

  - Fixed in: 7.2.1.0
  - Introduced in: 7.0.0.0
  - Support identifiers: DS-38344 SF#00655578
- The dsreplication tool has been fixed to work when the node being used to enable replication is currently out-of-sync with the topology master.

  - Fixed in: 7.2.1.0
  - Introduced in: 7.0.0.0
  - Support identifiers: DS-38335 SF#00655578
- Fixed two issues in which the server could have exposed some clear-text passwords in files on the server file system.

  * When creating an encrypted backup of the alarms, alerts, configuration, encryption settings, schema, tasks, or trust store backends, the password used to generate the encryption key (which may have been obtained from an encryption settings definition) could have been inadvertently written into

the backup descriptor. This problem does not affect local DB backends (like userRoot), the LDAP changelog backend, or the replication database.

* When running certain command-line tools with an argument instructing the tool to read a password from a file, the password contained in that file could have been written into the server's tool invocation log instead of the path to that file. Affected tools include backup, create-initial-config, create-initial-proxy-config, dsreplication, enter-lockdown-mode, export-ldif, import-ldif, ldappasswordmodify, leave-lockdown-mode, manage-tasks, manage-topology, migrate-ldap-schema, parallel-update, prepare-endpoint-server, prepare-external-server, realtime-sync, rebuild-index, re-encode-entries, reload-http-connection-handler-certificates, reload-index, remove-defunct-server, restore, rotate-log, and stop-server. Other tools are not affected. Also note that this only includes passwords contained in files that were provided as command-line arguments; passwords included in the tools.properties file, or in a file referenced from tools.properties, would not have been exposed.

In each of these cases, the files would have been written with permissions that make their contents only accessible to the system account used to run the server. Further, while administrative passwords may have been exposed in the tool invocation log, neither the passwords for regular users, nor any other data from their entries, should have been affected. We have introduced new automated tests to help ensure that such incidents do not occur in the future.

We recommend changing any administrative passwords you fear may have been compromised as a result of this issue. If you are concerned that the passphrase for an encryption settings definition may have been exposed, then we recommend creating a new encryption settings definition that is preferred for all subsequent encryption operations, exporting your data to LDIF, and re-importing so that it will be encrypted with the new key. You also may wish to re-encrypt or destroy any existing backups, LDIF exports, or other data encrypted with a compromised key, and you may wish to sanitize or destroy any existing tool invocation log files that may contain clear-text passwords.

  - Fixed in: 7.0.1.3
  - Introduced in: 7.0.0.0
  - Support identifiers: DS-38897 DS-38908
- The server can now detect an "out of file handles" situation on the operating system, and shut down to prevent running in an unreliable state.
  - Fixed in: 5.1.0.0
  - Introduced in: 2.1.0.0
  - Support identifiers: DS-12579 SF#2655
- Disabled support for SSLv3 by default in the LDAP, HTTP, and JMX connection handlers, and for replication communication. The recently-discovered POODLE vulnerability could potentially allow a network attacker to determine the plaintext behind an SSLv3-encrypted session, which would effectively negate the primary benefit of the encryption.

  SSLv3 was initially defined in 1996, but was supplanted by the release of the TLSv1 definition in 1999 (and subsequently by TLSv1.1 in 2006 and TLSv1.2 in 2008). These newer TLS protocols are not susceptible to the POODLE vulnerability, and the server has supported them (and preferred them over SSLv3) for many years. The act of disabling SSLv3 by default should not have any adverse effect on clients that support any of the newer TLS protocols. However, if there are any legacy client applications that attempt to communicate securely but do not support the newer TLS protocols, they should be updated to support the newer protocols. In the event that there are known clients that do not support any security protocol newer than SSLv3 and that cannot be immediately updated to support a newer protocol, SSLv3 support can be re-enabled using the newly-introduced allowed-insecure-tls-protocol global configuration property. However, since communication using SSLv3 can no longer be considered secure, it is strongly recommended that every effort be made to update all known clients still using SSLv3.

  It is possible to use the server access log to identify LDAP clients that use SSLv3 to communicate with the server. Whenever an LDAP client establishes a secure connection to the server, or whenever a client uses the StartTLS extended operation to secure an existing plaintext connection, the server will generate a SECURITY-NEGOTIATION access log message. The "protocol" element of a SECURITY-

NEGOTIATION access log message specifies the name of the security protocol that has been negotiated between the client and the server, and any SECURITY-NEGOTIATION messages with a protocol of "SSLv3" suggest that the associated client is vulnerable to the POODLE attack. In addition, if any connections are terminated for attempting to use the disallowed SSLv3 protocol, the access log message for that disconnect should include a message stating the reason for the termination.

  - Fixed in: 5.0.0.0
  - Introduced in: 2.1.0.0
  - Support identifiers: DS-11782
- Update the replication backlog health check so that if a problem is encountered while attempting to retrieve monitor information from a backend server, that server will only be classified as degraded rather than unavailable.
  - Fixed in: 4.5.0.0
  - Introduced in: 4.1.0.7
  - Support identifiers: DS-9726
- Fix a bug that allows users with expired passwords to change attributes in their own entry other than password.
  - Fixed in: 3.5.0.0
  - Introduced in: 3.2.0.0
  - Support identifiers: DS-6054
- Update the PingDirectoryProxy Server to apply access controls when processing the GetAuthorizationEntryRequestControl.
  - Fixed in: 3.5.0.0
  - Introduced in: 2.0.0.0
  - Support identifiers: DS-854

Resolved Issues

The following issues have been resolved with this release of the PingDirectoryProxy Server:

Ticket ID	Description
DS-40532	Added a logging-error-behavior property to the log publisher, periodic stats logger plugin, and monitor history plugin configuration that can be used to specify the behavior the server should exhibit if an error occurs while attempting logging-related processing. By default, the server will preserve its previous behavior of writing a message to standard error, but it can be configured to enter lockdown mode on a logging error, in which the server will report itself as unavailable and will only accept requests from accounts with the lockdown-mode privilege and only from clients communicating over a loopback interface.
DS-40681	Added a cache for password policies stored in user data rather than in the configuration. The cache will hold up to 500 policies by default, but the cache size can be configured (or the cache disabled) using the maximum-user-data-password-policies-to-cache property in the global configuration.

Ticket ID	Description
DS-41206	Fixed a memory leak when performing SCIM queries on the PingDirectory Server.

PingDirectoryProxy Server 8.0.0.0 Release Notes

Critical Fixes

This release of the PingDirectoryProxy Server addresses critical issues from earlier versions. Update all affected servers appropriately.

What's New

These are new features for this release of the PingDirectoryProxy Server:

- Better insight to server health and the performance of connected applications. Administrators can now push metrics to application insight and monitoring applications such as Splunk using two new methods. A new StatsD monitoring endpoint pushes metrics to StatsD-compatible services. Also, the periodic stats logger has been updated to use JSON-format log files, greatly simplifying the use of log forwarding tools like Splunk's Universal Forwarder or Elastic's FileBeats.
- Better support by Directory Proxy Server for orchestrated Directory Server clusters in automated environments like Kubernetes or AWS EC2 Auto Scaling. Administrators can connect the topology of a PingDirectoryProxy Server cluster to the topology of a PingDirectory Server cluster in order to automatically adapt to cluster changes within the PingDirectory Server cluster. This greatly simplifies the configuration management of PingDirectoryProxy Server when PingDirectory Server clusters are being added, removed, or replaced by automated infrastructure orchestrators.
- Use Server Profiles to reduce risk and improve consistency following the DevOps principle of infrastructure-as-code. Administrators can export the configuration of the PingDirectoryProxy Server server to a directory of text files called a Server Profile, track changes to these files in version control like Git, and install new instances of PingDirectoryProxy Server or update existing instances of Directory Proxy Server from a Server Profile. Server Profiles support variable substitution in order to remove the settings unique to each pre-production or production environment from the Server Profile that is stored in version control.

Known Issues/Workarounds

The following are known issues in the current version of the PingDirectoryProxy Server:

- The following are suggested solutions for problems with slow DNS:
  - Maintain a connection pool in the client app rather than opening new connections for each bind.
  - Add appropriate records, including PTR records, to DNS.
  - Add `options timeout:1` in the `/etc/resolv.conf` file and/or `options single-request`
  - If IPv6 requests specifically are causing issues, add `-Djava.net.preferIPv4Stack=true` to the `start-server.java-args` line in Directory Server's `config/java.properties` file, run `bin/dsjavaproperties`, and restart the server to stop the issuance of IPv6 PTR requests.
- Some server tools, such as `dsreplication`, `collect-support-data`, and `rebuild-index`, will fail with errors if they are run with an encrypted `tools.properties` file.

  Workaround: Add the `--noPropertiesFile` argument to the server tools to prevent them from pulling information from the encrypted file.
- The working directory value used by exec tasks is not implemented for recurring exec tasks.

- Deploying the Admin Console to an external container using JDK 11 requires downloading the following dependencies and making them available at runtime (for example, by copying them to the `WEB-INF/lib` directory of the exploded WAR file).
  - groupId:jakarta.xml.bind, artifactId:jakarta.xml.bind-api, version:2.3.2
  - groupId:org.glassfish.jaxb, artifactId:jaxb-runtime, version:2.3.2

  Workaround: Deploy the Console in an external container using JDK 8.

Resolved Issues

The following issues have been resolved with this release of the PingDirectoryProxy Server:

Ticket ID	Description
DS-17278	Added a `cn=Server Status Timeline,cn=monitor` monitor entry to track a history of the local server's last 100 status changes and their timestamps. Updated the LDAP external server monitor to include attributes tracking health check state changes for external servers. The new attributes include the number of times a health check transition has occurred, timestamps of the most recent transitions, and messages associated with the most recent transitions.
DS-37881	The PingFederate Access Token Validator will now refresh its cached value of the PingFederate server's token introspection endpoint. A new attribute, `endpoint-cache-refresh`, has been added to the PingFederate Access Token Validator, which will determine how often this refresh occurs.
DS-37955	To support multiple trace loggers, each trace logger now has its own resource key, which is shown in the `Resource` column in the output of `status`. This key allows multiple alarms, due to sensitive message types for multiple trace loggers.
DS-38053	The JWT Access Token Validator no longer requires a restart after a change to one of its signing certificates.
DS-38560	Updated `manage-profile replace-profile` to apply configuration changes directly, when possible. If the new server profile used by `replace-profile` has changed only the `dsconfig` batch files from the original profile, then only the `dsconfig` files are applied. If no changes are detected between profiles, `replace-profile` takes no action. If changes other than `dsconfig` are detected, the full `replace-profile` process is followed.
DS-38777	Added support for updating the server version during `manage-profile replace-profile`. The server must have been originally set up with a server profile.
DS-38832	Fixed an issue that could cause the server to leak a small amount of memory each time it failed to establish an LDAP connection to another server.

Ticket ID	Description
DS-38863	Updated the `manage-profile setup` subcommand to set a server's cluster name to match its instance name by default. This prevents servers in the same replication topology from being in the same cluster, reducing the risk of unintentionally overwriting parts of an existing server's configuration in a DevOps environment. The `--useDefaultClusterName` argument can be used to leave the cluster name unchanged.
DS-38867	Updated the PBKDF2 password storage scheme to add support for variants that use the 256-bit, 384-bit, and 512-bit SHA-2 digest algorithms. At present, the SHA-1 variant remains the default to preserve backward compatibility with older versions. Also, in accordance with the recommendations in NIST SP 800-63B, we have increased the default iteration count from 4096 to 10,000, and the default salt length from 64 bits to 128 bits.
DS-38869	Updated the `remove-defunct-server` tool's `--ignoreOnline` option. When using `--ignoreOnline` in a mixed-version environment, all servers must support the option.
DS-39155	Updated the default value for `maximum-degraded-missing-changes` to 1000000, and for `minimum-unavailable-missing-changes` to 100000. Servers that used the previous default values became unavailable prematurely, leading to outages.
DS-39176, DS-39308	Updated the Groovy scripting language version to 2.5.7. For a list of changes, visit groovy-lang.org and view the Groovy 2.5 release notes. As of this release, only the core Groovy runtime and the groovy-json module are bundled with the server. To deploy a Groovy-scripted Server SDK extension that requires a Groovy module not bundled with the server, such as groovy-xml or groovy-sql, download the appropriate jar file from *groovy-lang.org* and place it in the server's `lib/extensions` directory.
DS-39253	Added a `replace-certificate` tool, which can help an administrator replace the listener or inter-server certificate for a server instance.
DS-39321	Added support for PingDirectoryProxy Server to the `manage-profile` tool and its subcommands.

Ticket ID	Description
DS-39325	Removed the legacy product-specific scripts for starting and stopping the server. These include: ▪ `start-ds` and `stop-ds` for Directory Server ▪ `start-proxy` and `stop-proxy` for Directory Proxy Server ▪ `start-sync` and `stop-sync` for Data Sync Server ▪ start-metrics-engine and stop-metrics-engine for Data Metrics Server These legacy scripts had been deprecated for several releases in favor of the more general start-server and stop-server scripts, and they displayed a warning message about their upcoming removal if they were invoked. If you still have dependencies on these legacy product-specific scripts, you will need to update them to reference the general start-server and stop-server scripts instead. If it is not feasible to update these references immediately, you may create symbolic links that use the legacy script names and point at the start-server and stop-server scripts.
DS-39347	Fixed an issue where Delegated Admin would not work properly if the name of the REST Resource Type was not the same as the resource endpoint.
DS-39373	Preserve the privileges that are explicitly set on the admin user when migrating from the admin backend to the topology registry.
DS-39518	Fixed an issue in which escaped characters in schema extensions may not be handled properly. If used in attribute type constraints (such as `X-VALUE-REGEX`), this could cause unexpected or incorrect behavior.
DS-39525, DS-39526	Delegated Admin enhancements for constructed attributes. ▪ Allow a required attribute to be read-only if it is constructed. ▪ Add a configured list of "Update Constructed Attributes" on the REST resource type, similar to the "Post Create Constructed Attributes", so that constructed attributes can be updated when dependent attributes change. ▪ Handle constructed attributes which reference other constructed attributes.
DS-39592	HTTP External Servers have a new attribute, `ssl-cert-nickname`, which defines the alias of a specific certificate within their keystore to be used as a client certificate.
DS-39603	Fixed an issue where Server SDK extensions could not be configured by dsconfig batch files in the manage-profile tool.
DS-39626, DS-40357	The trace log publisher will now record an access token's scopes after the token is successfully validated.
DS-39654	Added support for the `--topologyFilePath` argument to the `manage-topology add-server` subcommand.
DS-39671	Updated the `manage-topology add-server` subcommand to require being run from the older server in a mixed-version environment.

Ticket ID	Description
DS-39693	Fixed an issue where Delegated Admin search results were truncated and invalid upon encountering a Directory entry containing a Boolean or Integer syntax attribute whose values were invalid because they did not conform to the appropriate syntax. With this fix, the offending values are omitted from the results and a warning message is logged to the server errors log.
DS-39715	Updated the Server SDK to add support for sending email messages.
DS-39762	Added support for a "generate password" extended operation that can be used to request that the server generate one or more passwords that may be suggested as possible values when creating a new user or changing the password for an existing user.
DS-39857	Added the StatsD monitoring endpoint. When the Stats Collector Plugin is enabled, this endpoint sends metric data from the server in StatsD format to the configured destination.
DS-39908	Added a new JVM-default trust manager provider that can be used to automatically trust any certificate signed by an authority included in the JVM's default set of trusted issuers. Also, updated other trust manager providers to offer an option to use the JVM-default trust addition to the trust that they normally provide.
DS-40114	Added a new `cn=Status Health Summary,cn=monitor` monitor entry that provides a summary of the server's current assessment of its health. This simplifies monitoring with third party tools that support retrieving monitoring data over JMX. The Periodic Stats Logger has also been updated to allow some of this monitoring information to be logged. No new information is logged by default.
DS-40249	Fixed an issue where an LDAP search across entry-balanced server sets sometimes returned 0 (success) even though all servers in one of the sets failed with a timeout. The search should return 52 (unavailable) in this situation.
DS-40252	Update the PingDirectoryProxy Server setup process to support joining an existing Ping Identity Platform topology.
DS-40255, DS-40256, DS-40257	Updated the fewest operations and failover load-balancing algorithms to support automatically discovering the set of backend Directory Server instances from the topology registry.
DS-40274	Updated the proxy to check if the proxy's backend has been sufficiently initialized prior to performing a health check.
DS-40354	Fixed a problem with `config-diff` when writing properties that span multiple lines using the `--prettyPrint` argument.
DS-40366	Fixed an issue where the server was attempting to connect by an IP address rather than a hostname when DNS lookup was successful.
DS-40377	Added support for logging to a JSON file in the Periodic Stats Logger Plugin.
DS-40517	Added metrics for status summary, replication database, and LDAP changelog to the Stats Collector Plugin.
DS-40543	Updated `manage-profile replace-profile` to copy the tool log file to the server being updated.

Ticket ID	Description
DS-40556	Added support for specifying a working directory for exec tasks.
DS-40730	Updated the `encrypt-file` tool to prevent using the same path for both the input file and the output file.
DS-40771	Added a `--duration` argument to collect-support-data. When used, only the log files covering the specified duration before the current time will be collected.

PingDataMetrics Release Notes

PingDataMetrics 8.2.0.8 release notes

The release notes for the 8.2.0.8 release of PingDataMetrics.

Critical fixes

This release of PingDataMetrics addresses critical issues from earlier versions. Update all affected servers appropriately.

No critical issues have been identified.

Resolved issues

The following issues have been resolved with this release of PingDataMetrics.

Ticket ID	Description
DS-45164	Updated Jetty Server to version 9.4.44.
DS-45813	Updated PingDirectory products to use Kafka v2.8.1.

PingDataMetrics Server 8.2.0.7 release notes

The release notes for the 8.2.0.7 release of PingDataMetrics Server.

Critical fixes

This release of PingDataMetrics Server addresses critical issues from earlier versions. Update all affected servers appropriately.

- Fixed an issue where secret keys under `cn=Topology,cn=config` could be lost when removing a server from the topology. When a server is removed via the `dsreplication disable` or **`remove-defunct-server`** tools, its secret keys will now be distributed among the remaining members of the topology. The keys from the rest of the topology will also be copied to the server being removed.

  The cipher secret keys in the topology that are affected by this change are used by reversible password storage schemes (except for AES256, which uses the encryption settings database). If you are using a reversible password storage scheme other than AES256, prior to this fix, you could lose access to keys that had been used for reversible password encryption when removing servers from the topology.

  ⓘ **Note:**

  Since this change only applies to the most recent version of **`remove-defunct-server`** and `dsreplication disable`, if you are removing a server from a multi-version topology, you should run that tool from the most recent version. In the past **`dsreplication`** and **`remove-defunct-server`**

could only be run from an older version, but now in the case of removing a server from the topology, they should be run from the most recent version in the topology. If you run the tool from an older server, it will not include this fix, and you might lose access to secret keys from servers that are removed from the topology.

- Fixed in: 8.2.0.7
- Introduced in: 7.0.0.0
- Support identifiers: DS-44591

Resolved Issues

The following issues have been resolved with this release of the Data Metrics Server.

Ticket ID	Description
DS-44591	Fixed an issue where secret keys under `cn=Topology,cn=config` could be lost when removing a server from the topology. When a server is removed via the `dsreplication disable` or **`remove-defunct-server`** tools, its secret keys will now be distributed among the remaining members of the topology. The keys from the rest of the topology will also be copied to the server being removed. The cipher secret keys in the topology that are affected by this change are used by reversible password storage schemes (except for AES256, which uses the encryption settings database). If you are using a reversible password storage scheme other than AES256, prior to this fix, you could lose access to keys that had been used for reversible password encryption when removing servers from the topology. ⓘ **Note:** Since this change only applies to the most recent version of **`remove-defunct-server`** and `dsreplication disable`, if you are removing a server from a multi-version topology, you should run that tool from the most recent version. In the past **`dsreplication`** and **`remove-defunct-server`** could only be run from an older version, but now in the case of removing a server from the topology, they should be run from the most recent version in the topology. If you run the tool from an older server, it will not include this fix, and you might lose access to secret keys from servers that are removed from the topology.

Ticket ID	Description
DS-45124	Removed `-XX:RefDiscoveryPolicy=1` from the default start-server Java arguments. In rare cases, this argument was related to segmentation faults in the JVM, especially when used with the G1 garbage collector.
DS-45190	Added support for the use of JDKs obtained through BellSoft.

PingDataMetrics Server 8.2.0.6 release notes

Critical fixes

This release of PingDataMetrics Server addresses critical issues from earlier versions. Update all affected servers appropriately.

No critical issues have been identified.

Resolved issues

The following issues have been resolved with this release of the DataMetrics Server.

Ticket ID	Description
DS-43632	Fixed an issue where the `format` field is omitted from the list of operational attribute schemas in the Directory REST API.
DS-44224	Addressed an issue where icons on the dashboards were not properly displayed.

PingDataMetrics 8.2.0.5 release notes

Critical fixes

This release of the PingDataMetrics Server addresses critical issues from earlier versions. Update all affected servers appropriately.

No critical issues have been identified.

Resolved issues

The following issues have been resolved with this release of the PingDataMetrics Server.

Ticket ID	Description
DS-44025	Fixed an issue where the server was incorrectly displaying an `Unknown vendor` warning when using JDKs obtained on Red Hat and Ubuntu systems.

PingDataMetrics Sever 8.2.0.2 release notes

Critical Fixes

This release of the PingDataMetrics Server addresses critical issues from earlier versions. Update all affected servers appropriately.

No critical issues have been identified

Resolved Issues

The following issues have been resolved with this release of the PingDataMetrics Server:

Ticket ID	Description
DS-43224	Made a generic OpenID Connect ID token validator available. This change allows single sign-on to the Administrative Console with OIDC providers other than just PingOne.
DS-43941	You can now specify that the Administrative Console use a custom truststore when evaluating OIDC provider certificates by using the oidc-trust-store-file and oidc-trust-store-type settings. Also, you can set the console to skip hostname and/or certification verification through the oidc-strict-hostname-verification and oidc-trust-all configuration settings.

PingDataMetrics Server 8.2.0.1 release notes

Critical Fixes

This release of the Data Metrics Server addresses critical issues from earlier versions. Update all affected servers appropriately.

No critical issues have been identified

Data Metrics Server 8.2.0.0 Release Notes

Upgrade Considerations

Important considerations for upgrading to this version of the Data Metrics Server

- If you have upgraded a server that is in a cluster (i.e., has a cluster name set in the Server Instance configuration object) to version 8.1, you will not be able to make cluster configuration changes until all servers with the same cluster name have been upgraded to version 8.1. If needed, you could create temporary clusters based on server versions and modify each of the servers' cluster name appropriately to minimize the impact while you are upgrading.
- Updated the "delay bind response" failure lockout action to provide an option to delay the response to bind requests initiated by non-LDAP clients (for example, when using HTTP basic authentication). This option is disabled by default because delaying the bind response for non-LDAP clients may require temporarily blocking the thread used to process the request, which could increase the risk of a denial-of-service attack. To help mitigate this risk, if you enable delayed bind responses for non LDAP clients, we recommend that you also increase the number of request handler threads for all enabled HTTP connection handlers.

- Updated setup to create a second encryption settings definition if data encryption is enabled. It will continue to create a definition for 128-bit AES encryption for use as the preferred definition to preserve backward compatibility with existing servers in the topology, but it will now also generate a definition for 256-bit AES encryption. The 256-bit AES definition may become the preferred definition in a future release, but you can use it now by first ensuring that any existing instances are updated to contain the new definition (with the "encryption-settings export" and "encryption-settings import" commands) and then making it the preferred definition (with "encryption-settings set-preferred") in all instances.

What's New

These are new features for this release of the Data Metrics Server

- In an ongoing effort to improve the use of containers for PingDirectory, several features have been implemented:

  - The --outputFile option has been added to the collect-support-data tool. You can now specify either a path, a file name, or a path and file name for the resulting CSD file. This means an administrator can run the collect-support-data tool and put the output file into a directory outside of the container, allowing access to the file without having to actually connect to the container.

  - The collect-support-data tool can now be run as a recurring task. Recurring tasks can be created using the Administration console which means that administrators do not have to connect to the container in order to run the tool.

  - A Collect Support Tool Extended Operation has been added allowing LDAP clients to initiate the collect-support-data tool and to receive the output of the request. The LDAP SDK has been updated to support this, and the --remoteServer added to the collect-support-data tool can be used to send the request to another server. In other words, you can now run collect-support-data on the command line and reference another server, possibly in a container, and retrieve the output file remotely.
- PingDirectory has a Consent REST API that allows users to create and store consents. A new feature now allows users to search for consents that have been granted to them by another party.

Known Issues/Workarounds

The following are known issues in the current version of the Data Metrics Server

- Several known issues can occur when you use the Administrative Console with Tomcat 9.0.31. You can resolve these issues by upgrading to Tomcat 9.0.33 or later.
- If you use the create-systemd-script tool to create a forking systemd service, the service is stopped by the "systemctl stop ping-directory.service" command. At that time, you can see the status using the "systemctl status ping-directory.service" command. That status might contain an indication of failure: "Active: failed (Result: exit-code)". This error has to do with the way the service exits. It is harmless.

Resolved Issues

The following issues have been resolved with this release of the Data Metrics Server:

Ticket ID	Description
DS-5143, DS-11035	Updated support for logging access and error log messages to a syslog server. While the server previously supported logging these messages to a syslog server (through the "syslog-based access log publisher" and "syslog-based error log publisher" logger implementations), these loggers used an older version of the syslog protocol (described in RFC 3164) and only offered support for communicating over UDP. These loggers are still available for legacy backward compatibility, but we now also offer new "syslog text access log publisher" and "syslog text error log publisher" implementations that use a newer version of the syslog protocol (syslog version 1, described in RFC 5424) and support communicating over UDP or the more reliable TCP. When using TCP, it is also possible to encrypt communication with TLS and to configure multiple servers for better redundancy. These loggers use the same space-delimited text format as the former loggers. We also offer new "syslog JSON access log publisher" and "syslog JSON error log publisher" implementations that offer the same set of capabilities, but that format the message text as JSON objects, which can be more easily parsed by third-party software.
DS-10320, DS-12550, DS-12551, DS-12552, DS-42116, DS-42162, DS-42179, DS-42222, DS-42223, DS-42224, DS-42225, DS-42416, DS-42437	Added a config/sample-dsconfig-batch-files directory with set of well commented dsconfig batch files that may be useful in enabling or configuring a variety of features in the server.
DS-10775	Updated the dictionary password validator to support additional options: ▪ It can now ignore non-alphabetic characters that appear at the beginning or end of the password before checking the dictionary. ▪ It can strip characters of diacritical marks, including accents, cedillas, circumflexes, diaereses, tildes, and umlauts, before checking the dictionary. If this option is used, then any character with such a mark will be replaced with a base version of the character without that mark (for example, a lowercase letter n with a tilde over it would be replaced with just a lowercase letter n). ▪ You can define maps with information about character substitutions to use for checking alternative versions of the provided password. For example, if you indicate that "0" might map to "o", "1" or "!" might map to "i", "7" might map to "t", and "3" might map to "e", then the validator can reject a proposed password of "pr0h1b!73d" if the dictionary contains the word "prohibited". ▪ It can reject a proposed password if a value from the provided dictionary makes up more than a specified percentage of that password.

Ticket ID	Description
DS-11524, DS-41860, DS-42112	Added support for new administrative alert types. ▪ We have added a new admin alert account status notification handler, which can generate administrative alerts whenever an applicable account status notification is generated within the server. For example, this account status notification handler can be added to the root password policy to generate an alert whenever a root user's password is updated or their account is locked as a result of too many failed authentication attempts. A separate alert type has been defined for each account status notification type. ▪ We have added a new "privilege-assigned" administrative alert that can be raised whenever a new entry is added or an existing entry is updated to include one or more privileges. ▪ We have added a new "insecure-request-rejected" administrative alert that can be raised whenever the server rejects a request as a result of the reject-insecure-requests global configuration property.
DS-13853	Added support for the OAUTHBEARER SASL mechanism (as described in RFC 7628) to allow LDAP clients to authenticate with OAuth 2.0 bearer tokens.
DS-15848, DS-42360	Added support for invoking a specified set of password validators during bind operations. If the password used to authenticate fails to satisfy one or more of the configured validators, the bind attempt can be rejected, the user can be forced to change their password, or the server can generate an account status notification to take some alternative action (for example, notifying the end user or server administrators).
DS-15864	Replaced the ldappasswordmodify tool with a new version that offers more functionality, including support for additional controls, support for multiple password change methods (the password modify extended operation, a regular LDAP modify operation, or an Active Directory-specific modify operation), and the ability to generate the new password on the client.
DS-17903	Updated setup to provide a --populateToolPropertiesFile argument that will allow it to populate the config/tools.properties file with default values for command-line tool arguments. If requested, properties will be provided for the server address, port, and communication security, and may also include a default bind DN and optionally a bind password. When running setup interactively, it will now prompt to determine which properties (if any) should be populated in the properties file.
DS-38110	Updated the System Information monitor with an "isDocker" attribute to identify if the server is running in a Docker container.
DS-38118, DS-42495	Made several updates related to the server's handling of data written to standard output and standard error: ▪ The server can now be configured to rotate the logs/server.out file once it reaches a given size, and it will retain a configurable number of those log files. By default, the server will rotate the file once it reaches 100 megabytes and will keep up to ten files. ▪ To better facilitate capturing log data in containerized environments, the server now supports writing JSON-formatted access and error log messages to the JVM's original standard output and error streams (which will be separate from the server.out file when the server is started with the --nodetach argument). ▪ It is now possible to prevent the server from logging messages during startup in non-JSON format. It is also possible to prevent messages about administrative alerts from being written to standard error, or to write those messages in JSON format. These options are especially useful when using JSON-based logging to the console in no-detach mode, as they can help ensure that everything written to standard output and standard error will be formatted as JSON objects.

Ticket ID	Description
DS-38868	Updated setup to create a second encryption settings definition if data encryption is enabled. It will continue to create a definition for 128-bit AES encryption for use as the preferred definition to preserve backward compatibility with existing servers in the topology, but it will now also generate a definition for 256-bit AES encryption. The 256-bit AES definition may become the preferred definition in a future release, but you can use it now by first ensuring that any existing instances are updated to contain the new definition (with the "encryption-settings export" and "encryption-settings import" commands) and then making it the preferred definition (with "encryption-settings set-preferred") in all instances.
DS-39789	Updated the JVM memory usage monitor provider to fix an issue that could prevent the monitor from reporting the total amount of memory held by all memory consumers. Also, fixed an issue that could cause the memory-consumer attribute to use an incomplete message for consumers without a defined maximum size and added an additional memory-consumer-json attribute whose values are JSON objects with data that can be more easily extracted by automated processes.
DS-40650	Updated the collect-support-data tool to make it possible to specify how much data should be captured from the beginning and end of each log file to include in the support data archive. You can also specify the capture size when invoking the tool through an administrative task, recurring task, or extended operation.
DS-40967	Eliminated a misleading error message that could be logged at startup if the server was configured with one or more ACIs that only apply when using specific SASL mechanisms.
DS-41989	Fixed an issue that could result in duplicate column headers being produced by the Periodic Stats Logger, even when the header-prefix-per-column attribute was set to true.
DS-42045	Updated the Stats Collector Plugin with a new generate-collector-files configuration property. When using this plugin with the Data Metrics Server, leave this property at its default value of true. When using the plugin exclusively for providing metrics to one or more StatsD Monitoring Endpoints, set this property to false to prevent unnecessary I/O.
DS-42059, DS-42060	Updated setup to add options for improving communication security: ▪ Non-interactive setup now offers a --rejectInsecureRequests argument that will configure the server to reject any request received over a connection that is not encrypted with SSL or StartTLS. ▪ Non-interactive setup now offers a --rejectUnauthenticatedRequests argument that will configure the server to reject any request received over a connection that is not authenticated (or that is authenticated as the anonymous user). ▪ Interactive setup now allows you to configure the server with the LDAP connection handler disabled (which was already an option when using non-interactive setup), or enabled but only for communication encrypted with StartTLS. The --rejectInsecureRequests and --rejectUnauthenticatedRequests arguments can also be used with manage-profile by including them in the setup-arguments.txt file of the server profile.

Ticket ID	Description
DS-42061	Updated the interactive command-line tool framework to prefer establishing secure LDAP connections over insecure connections. Previously, when prompting for the information needed to establish a connection, the default option was to create an unencrypted LDAP connection. Now, tools default to creating an SSL-encrypted connection if the server supports it, or to creating a StartTLS-encrypted connection if that is available but SSL is not. Tools also default to using streamlined settings when establishing secure connections. Previously, they would always prompt about how to determine whether the server's certificate chain should be trusted. When using the streamlined settings, the tools only prompt about certificates that cannot automatically be considered trusted using information in the JVM's default trust store, the server's default trust store (config/truststore), or the server's topology registry.
DS-42062	Updated the root password policy so that LDAP bind responses for root users and topology administrators will be delayed by one second after five consecutive failed authentication attempts.
DS-42063	Updated the "delay bind response" failure lockout action to provide an option to delay the response to bind requests initiated by non-LDAP clients (for example, when using HTTP basic authentication). This option is disabled by default because delaying the bind response for non-LDAP clients may require temporarily blocking the thread used to process the request, which could increase the risk of a denial-of-service attack. To help mitigate this risk, if you enable delayed bind responses for non-LDAP clients, we recommend that you also increase the number of request handler threads for all enabled HTTP connection handlers.
DS-42115	Updated the server's command-line tool framework to make it easier and more convenient to communicate with the server over a secure connection when no trust-related arguments are provided. Most non-interactive tools will now check the server's default trust store, the topology registry, and the JVM's default trust store to see if the presented certificate chain can be automatically trusted without the need to prompt the user. If the presented chain cannot be automatically trusted, the user may be interactively prompted to determine whether it should be trusted.
DS-42279	Updated the server to require a minimum key size of 2048 bits when negotiating a TLS cipher suite that uses ephemeral Diffie-Hellman key exchange.
DS-42298	Replaced the ldifsearch, ldifmodify, and ldif-diff command-line tools with more full-featured and robust implementations.
DS-42331	Replaced the ldapcompare tool with a new version that offers more functionality, including support for multiple compare assertions, following referrals, additional controls, and multiple output formats (including tab-delimited text, CSV, and JSON).
DS-42347	Updated the server to use /dev/urandom (on non-Windows systems where that path exists and is readable) instead of /dev/random as the primary source for secure random data. Attempts to read from /dev/random can block if the underlying system does not have sufficient entropy, which can have a severe adverse effect on performance. Reads from /dev/urandom will not block, and the data that it provides is no less secure than data from /dev/random in any way that matters for the server.
DS-42504	Updated manage-profile replace-profile to set encryption settings definitions defined in the newer server profile as preferred in the encryption settings db.
DS-42547	Fixed an issue where **`manage-profile generate-profile`** would print `null` as the generated profile directory when writing to an existing directory.

Ticket ID	Description
DS-42609	Fixed an issue in which the Directory REST API could fail to decode certain credentials when using basic authentication.
DS-42632	Added support for creating or importing a key pair configuration object using an elliptic curve (EC) key algorithm. You can use this to designate the encryption key pair for a JWT access token validator that handles EC-encrypted access tokens.
DS-42634	The JWT Access Token Validator can now validate JWT access tokens signed using the elliptic curve digital signature algorithms ES256, ES384, and ES512.
DS-42635	The JWT Access Token Validator can now validate JWT access tokens encrypted using elliptic curve cryptographic algorithms. The following key encryption algorithms are now supported in addition to RSA-OAEP: ECDH-ES, ECDH-ES+A128KW, ECDH-ES+A192KW, and ECDH-ES +A256KW. To support best practices for JWT security, you must now also configure the JWT Access Token Validator with explicit allow lists for key encryption and content encryption algorithms. For backward compatibility, the key encryption allow list defaults to RSA-OAEP, while the content encryption allow list defaults to A128CBC-HS256, A192CBC-HS384, and A256CBC-HS512. We recommend setting both allow lists to the strict minimum set of algorithms needed by the Access Token Validator.
DS-42651	Updated the **`manage-profile replace-profile`** subcommand to better support updating the server's keystore and truststore files. When using the `--generateSelfSignedCertificate` argument in a server profile's `setup-arguments.txt` file, the server maintains the original keystore and truststore files during **`replace-profile`**. Otherwise, **`replace-profile`** uses the keystore and truststore specified in the profile's `setup-arguments.txt` file.
DS-42667	Updated the server to set a unique cluster name when started for the first time.
DS-42669, DS-42748	Updated the online dsconfig step of the manage-profile replace-profile subcommand to support getting LDAP connection arguments from a tools.properties file on the server being updated. Fixed an issue where boolean LDAP connection arguments like --useSSL and --trustAll would cause manage-profile replace-profile to fail when applying dsconfig online.
DS-42681, DS-42684	Performance statistics generated by the Sideband API can now be published by the Periodic Stats Logger. To enable this, use the "included-http-servlet-stat" property of the Periodic Stats Logger.
DS-42673	Updated the **`manage-profile setup`** subcommand to fail if the **`start-server`** command has a non-zero exit code.
DS-42687	Upgrade to Jetty 9.4.30.
DS-42740	Fixed an issue where the **`dsconfig list`** subcommand would not display requested properties.
DS-42749	To support best practices for JWT security, you must now configure the JWT Access Token Validator with an explicit list of the JWT signing algorithms that it accepts. For backward compatibility, this list defaults to the RSA signing algorithms RS256, RS384, and RS512, but we recommend setting this list to the strict minimum set of signing algorithms needed by the Access Token Validator.

Ticket ID	Description
DS-42751	Added new override-status-code and additional-response-contents attributes to the Availability State HTTP Servlet Extension. These new attributes can be used to customize the response code and JSON response body of the servlet.
DS-42861	Updated the **`manage-profile`** tool logs to include the duration of each step the tool takes. The new `--verbose` argument can also be used to display timing information in the tool's console output.
DS-42886	Updated non-interactive **`setup`** (including **`manage-profile setup`**) to allow the password for the initial root user to be provided in pre-encoded form using the PBKDF2, SSHA256, SSHA384, or SSHA512 password storage scheme. This eliminates the need to have access to the clear-text password when setting up the server.
DS-42926	Fixed an issue where Ping Directory products configured to run as Microsoft Windows services were sometimes unable to automatically restart following an unplanned reboot, due to errors reading a corrupted server status file.
DS-42952	For Windows only, there can be a hang on start when global configuration property `startup-error-logger-output-location` is set to values that contain `standard-error`. For Windows only, `standard-error` values are silently mapped to equivalent `standard-output` values.
DS-42963	Updated the **`manage-profile generate-profile`** subcommand to ignore files larger than 100 megabytes when generating a server profile. Fixed an issue where many large files in the server root could cause the tool to run out of memory.
DS-43376	Updated log publisher logic to reduce the amount of CPU that the server consumes when it is idle.
DS-43480	Updated the system information monitor provider to restrict the set of environment variables that can be included. Previously, the monitor entry included information about all defined environment variables, as that information can be useful for diagnostic purposes. However, some deployments might include credentials, secret keys, or other sensitive information in environment variables, and that should not be exposed in the monitor. The server now only includes values from a predefined set of environment variables that are expected to be the most useful for troubleshooting problems and that are not expected to contain sensitive information.
DS-43517	Updated the jose4j library used for JWT signing and encryption to version 0.7.2.
DS-43651	The Security Guide is now available online at pingidentity.com. The guide has been removed from the server packaging.

Critical fixes

Critical Fixes

This release of PingDataMetrics Server addresses critical issues from earlier versions. Update all affected servers appropriately.

- Addressed an issue that could lead to slow, off-heap memory growth. This only occurred on servers whose cn=Version,cn=monitor entry was retrieved frequently.
  - Fixed in: 8.1.0.0
  - Introduced in: 5.2.0.0
  - Support identifiers: DS-41301

- Fixed a memory leak when performing SCIM queries on the Directory Server.
  - Fixed in: 8.1.0.0
  - Introduced in: 7.2.0.0
  - Support identifiers: DS-41206 SF#00681395
- Fixed a memory leak when performing SCIM queries on PingDataMetrics Server.
  - Fixed in: 8.0.0.1

    Introduced in: 7.2.0.0

    Support identifiers: DS-41206 SF#00681395
- Addressed an issue that could lead to slow off-heap memory growth. This only occurred on servers whose cn=Version,cn=monitor entry was retrieved frequently.
  - Fixed in: 8.0.0.1

    Introduced in: 5.2.0.0

    Support identifiers: DS-41301
- The following enhancements were made to the topology manager to make it easier to diagnose connection errors:

  - Added monitoring information for all the failed outbound connections (including the time since it's been failing and the last error message seen when the failure occurred) from a server to one of its configured peers and the number of failed outbound connections.

  - Added alarms/alerts for when a server fails to connect to a peer server within a configured grace period.
  - Fixed in: 7.3.0.0
  - Introduced in: 7.0.0.0
  - Support identifiers: DS-38334 SF#00655578
- The topology manager will now raise a mirrored-subtree-manager-connection-asymmetry alarm when a server is able to establish outbound connections to its peer servers, but those peer servers are unable to establish connections back to the server within the configured grace period. The alarm is cleared as soon as there is connection symmetry.
  - Fixed in: 7.3.0.0
  - Introduced in: 7.0.0.0
  - Support identifiers: DS-38344 SF#00655578
- The dsreplication tool has been fixed to work when the node being used to enable replication is currently out-of-sync with the topology master.
  - Fixed in: 7.3.0.0
  - Introduced in: 7.0.0.0
  - Support identifiers: DS-38335 SF#00655578
- Fixed two issues in which the server could have exposed some clear-text passwords in files on the server file system.

  * When creating an encrypted backup of the alarms, alerts, configuration, encryption settings, schema, tasks, or trust store backends, the password used to generate the encryption key (which may have been obtained from an encryption settings definition) could have been inadvertently written into the backup descriptor. This problem does not affect local DB backends (like userRoot), the LDAP changelog backend, or the replication database.

  * When running certain command-line tools with an argument instructing the tool to read a password from a file, the password contained in that file could have been written into the server's tool invocation log instead of the path to that file. Affected tools include backup, create-initial-config, create-initial-proxy-config, dsreplication, enter-lockdown-mode, export-ldif, import-ldif, ldappasswordmodify, leave-lockdown-mode, manage-tasks, manage-topology, migrate-ldap-schema, parallel-update, prepare-

endpoint-server, prepare-external-server, realtime-sync, rebuild-index, re-encode-entries, reload-http-connection-handler-certificates, reload-index, remove-defunct-server, restore, rotate-log, and stop-server. Other tools are not affected. Also note that this only includes passwords contained in files that were provided as command-line arguments; passwords included in the tools.properties file, or in a file referenced from tools.properties, would not have been exposed.

In each of these cases, the files would have been written with permissions that make their contents only accessible to the system account used to run the server. Further, while administrative passwords may have been exposed in the tool invocation log, neither the passwords for regular users, nor any other data from their entries, should have been affected. We have introduced new automated tests to help ensure that such incidents do not occur in the future.

We recommend changing any administrative passwords you fear may have been compromised as a result of this issue. If you are concerned that the passphrase for an encryption settings definition may have been exposed, then we recommend creating a new encryption settings definition that is preferred for all subsequent encryption operations, exporting your data to LDIF, and re-importing so that it will be encrypted with the new key. You also may wish to re-encrypt or destroy any existing backups, LDIF exports, or other data encrypted with a compromised key, and you may wish to sanitize or destroy any existing tool invocation log files that may contain clear-text passwords.

- Fixed in: 7.3.0.0
- Introduced in: 7.0.0.0
- Support identifiers: DS-38897 DS-38908

- The following enhancements were made to the topology manager to make it easier to diagnose the connection errors:

  - Added monitoring information for all the failed outbound connections (including the time since it's been failing and the last error message seen when the failure occurred) from a server to one of its configured peers and the number of failed outbound connections.

  - Added alarms/alerts for when a server fails to connect to a peer server within a configured grace period.

  - Fixed in: 7.2.1.0
  - Introduced in: 7.0.0.0
  - Support identifiers: DS-38334 SF#00655578
- The topology manager will now raise a mirrored-subtree-manager-connection-asymmetry alarm when a server is able to establish outbound connections to its peer servers, but those peer servers are unable to establish connections back to the server within the configured grace period. The alarm is cleared when connection symmetry is achieved.

  - Fixed in: 7.2.1.0
  - Introduced in: 7.0.0.0
  - Support identifiers: DS-38344 SF#00655578
- The dsreplication tool has been fixed to work when the node being used to enable replication is currently out-of-sync with the topology master.

  - Fixed in: 7.2.1.0
  - Introduced in: 7.0.0.0
  - Support identifiers: DS-38335 SF#00655578
- Fixed two issues in which the server could have exposed some clear-text passwords in files on the server file system.

  * When creating an encrypted backup of the alarms, alerts, configuration, encryption settings, schema, tasks, or trust store backends, the password used to generate the encryption key (which may have been obtained from an encryption settings definition) could have been inadvertently written into

the backup descriptor. This problem does not affect local DB backends (like userRoot), the LDAP changelog backend, or the replication database.

* When running certain command-line tools with an argument instructing the tool to read a password from a file, the password contained in that file could have been written into the server's tool invocation log instead of the path to that file. Affected tools include backup, create-initial-config, create-initial-proxy-config, dsreplication, enter-lockdown-mode, export-ldif, import-ldif, ldappasswordmodify, leave-lockdown-mode, manage-tasks, manage-topology, migrate-ldap-schema, parallel-update, prepare-endpoint-server, prepare-external-server, realtime-sync, rebuild-index, re-encode-entries, reload-http-connection-handler-certificates, reload-index, remove-defunct-server, restore, rotate-log, and stop-server. Other tools are not affected. Also note that this only includes passwords contained in files that were provided as command-line arguments; passwords included in the tools.properties file, or in a file referenced from tools.properties, would not have been exposed.

In each of these cases, the files would have been written with permissions that make their contents only accessible to the system account used to run the server. Further, while administrative passwords may have been exposed in the tool invocation log, neither the passwords for regular users, nor any other data from their entries, should have been affected. We have introduced new automated tests to help ensure that such incidents do not occur in the future.

We recommend changing any administrative passwords you fear may have been compromised as a result of this issue. If you are concerned that the passphrase for an encryption settings definition may have been exposed, then we recommend creating a new encryption settings definition that is preferred for all subsequent encryption operations, exporting your data to LDIF, and re-importing so that it will be encrypted with the new key. You also may wish to re-encrypt or destroy any existing backups, LDIF exports, or other data encrypted with a compromised key, and you may wish to sanitize or destroy any existing tool invocation log files that may contain clear-text passwords.

- Fixed in: 7.0.1.3
- Introduced in: 7.0.0.0
- Support identifiers: DS-38897 DS-38908

Release Notes Archive

Release notes for previous versions of Ping Data Metrics Server are included for reference.

PingDataMetrics Server 8.1.0.6 release notes
The release notes for the 8.1.0.6 release of PingDataMetrics Server.

Critical fixes

This release of the Data Metrics Server addresses critical issues from earlier versions. Update all affected servers appropriately.

No critical issues have been identified.

Resolved issues

The following issues have been resolved with this release of the Data Metrics Server.

Ticket ID	Description
DS-44224	Addressed an issue where icons on the dashboards were not properly displayed.

PingDataMetrics 8.1.0.5 release notes

Critical fixes

This release of the Data Metrics Server addresses critical issues from earlier versions. Update all affected servers appropriately.

No critical issues have been identified.

Resolved issues

The following issues have been resolved with this release of the Data Metrics Server.

Ticket ID	Description
DS-43288	Updated setup and the replace-certificate tool to improve the way we generate self-signed certificates and certificate signing requests to make them more palatable to clients. To reduce the frequency with which administrators had to replace self-signed certificates, we previously used a very long lifetime for self-signed certificates generated by setup or the replace-certificate tool. However, some clients (especially web browsers and other HTTP clients) have started more strenuously objecting to certificates to long lifetimes, so we now generate self-signed certificates with a one-year validity period. The inter-server certificate (which is used internally within the server and does not get exposed to normal clients) is still created with a twenty-year lifetime. Also, the replace-certificate tool's interactive mode has been updated to improve the process that it uses to obtain information to include in the subject DN and subject alternative name extension for self-signed certificates and certificate signing requests. The following changes have been made in accordance with CA/Browser Forum guidelines: * When selecting the subject DN for the certificate, we listed a number of common attributes that may be used, including CN, OU, O, L, ST, and C. We previously indicated that CN attribute was recommended. We now also indicate that the O and C attributes are recommended as well. * When obtaining the list of DNS names to include in the subject alternative name extension, we previously suggested all names that we could find associated with interfaces on the local system. In many cases, we now omit non-qualified names and names that are associated with loopback interfaces. We will also warn about any attempts to add unqualified or invalid names to the list. * When obtaining the list of IP addresses to include in the subject alternative name extension, we previously suggested all addresses associated with all network interfaces on the system. We no longer suggest any IP addresses associated with loopback interfaces, and we no longer suggest any IP addresses associated in IANA-reserved ranges (for example, addresses reserved for private-use networks). The tool will now warn about attempts to add these addresses for inclusion in the subject alternative name extension.

Ticket ID	Description
DS-44316	Reduced the JVM memory requirements for many command line tools. This avoids memory pressure when multiple tools, such as a scheduled collect-support-data task, are run concurrently to the server process. For most tools, the initial heap size has been reduced to 128 MB, and for certain tools the maximum heap size has capped at 512 MB. On systems with larger amounts of memory, these tools previously were allotted unnecessarily large heaps. The maximum heap size has not been reduced for any tool that especially benefits from having more memory.

Data Metrics Server 8.1.0.2 Release Notes

Critical Fixes

This release of the Data Metrics Server addresses critical issues from earlier versions. Update all affected servers appropriately.

- Addressed an issue that could lead to slow, off-heap memory growth. This only occurred on servers whose cn=Version,cn=monitor entry was retrieved frequently.
  - Fixed in: 8.1.0.0
  - Introduced in: 5.2.0.0
  - Support identifiers: DS-41301
- Fixed a memory leak when performing SCIM queries on the Directory Server.
  - Fixed in: 8.1.0.0
  - Introduced in: 7.2.0.0
  - Support identifiers: DS-41206 SF#00681395
- The following enhancements were made to the topology manager to make it easier to diagnose connection errors:
  - Added monitoring information for all the failed outbound connections (including the time since it has been failing and the last error message seen when the failure occurred) from a server to one of its configured peers and the number of failed outbound connections.
  - Added alarms/alerts for when a server fails to connect to a peer server within a configured grace period.
  - Fixed in: 7.3.0.0
  - Introduced in: 7.0.0.0
  - Support identifiers: DS-38334 SF#00655578
- The topology manager will now raise a mirrored-subtree-manager-connection-asymmetry alarm when a server is able to establish outbound connections to its peer servers, but those peer servers are unable to establish connections back to the server within the configured grace period. The alarm is cleared as soon as there is connection symmetry.
  - Fixed in: 7.3.0.0
  - Introduced in: 7.0.0.0
  - Support identifiers: DS-38344 SF#00655578

- The dsreplication tool has been fixed to work when the node being used to enable replication is currently out-of-sync with the topology master.
  - Fixed in: 7.3.0.0
  - Introduced in: 7.0.0.0
  - Support identifiers: DS-38335 SF#00655578
- Fixed two issues in which the server could have exposed some clear-text passwords in files on the server file system.
  - Fixed in: 7.3.0.0
  - Introduced in: 7.0.0.0
  - Support identifiers: DS-38897 DS-38908
- The following enhancements were made to the topology manager to make it easier to diagnose the connection errors:
  - Fixed in: 7.2.1.0
  - Introduced in: 7.0.0.0
  - Support identifiers: DS-38334 SF#00655578
- The topology manager will now raise a mirrored-subtree-manager-connection-asymmetry alarm when a server is able to establish outbound connections to its peer servers, but those peer servers are unable to establish connections back to the server within the configured grace period. The alarm is cleared when connection symmetry is achieved.
  - Fixed in: 7.2.1.0
  - Introduced in: 7.0.0.0
  - Support identifiers: DS-38344 SF#00655578
- The dsreplication tool has been fixed to work when the node being used to enable replication is currently out-of-sync with the topology master.
  - Fixed in: 7.2.1.0
  - Introduced in: 7.0.0.0
  - Support identifiers: DS-38335 SF#00655578
- Fixed two issues in which the server could have exposed some clear-text passwords in files on the server file system.
  - Fixed in: 7.0.1.3
  - Introduced in: 7.0.0.0
  - Support identifiers: DS-38897 DS-38908
- The server can now detect an "out of file handles" situation on the operating system, and shut down to prevent running in an unreliable state.
  - Fixed in: 5.1.0.0
  - Introduced in: 2.1.0.0
  - Support identifiers: DS-12579 SF#2655
- Disabled support for SSLv3 by default in the LDAP, HTTP, and JMX connection handlers, and for replication communication. The recently-discovered POODLE vulnerability could potentially allow a network attacker to determine the plaintext behind an SSLv3-encrypted session, which would effectively negate the primary benefit of the encryption.
  - Fixed in: 5.0.0.0
  - Introduced in: 2.1.0.0
  - Support identifiers: DS-11782

Resolved Issues

The following issues have been resolved with this release of the Data Metrics Server:

Ticket ID	Description
DS-40828	Fixed an issue where some state associated with a JMX connection was not freed after the connection was closed. This led to a slow memory leak in servers that were monitored by an application that created a new JMX connection each polling interval.
DS-42687	Upgrade to Jetty 9.4.30

PingDataMetrics Server 8.1 Release Notes

Critical Fixes

This release of PingDataMetrics Server addresses critical issues from earlier versions. Update all affected servers appropriately.

- Addressed an issue that could lead to slow, off-heap memory growth. This only occurred on servers whose cn=Version,cn=monitor entry was retrieved frequently.
  - Fixed in: 8.1.0.0
  - Introduced in: 5.2.0.0
  - Support identifiers: DS-41301
- Fixed a memory leak when performing SCIM queries on the Directory Server.
  - Fixed in: 8.1.0.0
  - Introduced in: 7.2.0.0
  - Support identifiers: DS-41206 SF#00681395
- Fixed a memory leak when performing SCIM queries on PingDataMetrics Server.
  - Fixed in: 8.0.0.1

    Introduced in: 7.2.0.0

    Support identifiers: DS-41206 SF#00681395
- Addressed an issue that could lead to slow off-heap memory growth. This only occurred on servers whose cn=Version,cn=monitor entry was retrieved frequently.
  - Fixed in: 8.0.0.1

    Introduced in: 5.2.0.0

    Support identifiers: DS-41301
- The following enhancements were made to the topology manager to make it easier to diagnose connection errors:

  - Added monitoring information for all the failed outbound connections (including the time since it's been failing and the last error message seen when the failure occurred) from a server to one of its configured peers and the number of failed outbound connections.

  - Added alarms/alerts for when a server fails to connect to a peer server within a configured grace period.

  - Fixed in: 7.3.0.0
  - Introduced in: 7.0.0.0
  - Support identifiers: DS-38334 SF#00655578
- The topology manager will now raise a mirrored-subtree-manager-connection-asymmetry alarm when a server is able to establish outbound connections to its peer servers, but those peer servers are unable

to establish connections back to the server within the configured grace period. The alarm is cleared as soon as there is connection symmetry.

  - Fixed in: 7.3.0.0
  - Introduced in: 7.0.0.0
  - Support identifiers: DS-38344 SF#00655578
- The dsreplication tool has been fixed to work when the node being used to enable replication is currently out-of-sync with the topology master.
  - Fixed in: 7.3.0.0
  - Introduced in: 7.0.0.0
  - Support identifiers: DS-38335 SF#00655578
- Fixed two issues in which the server could have exposed some clear-text passwords in files on the server file system.

  * When creating an encrypted backup of the alarms, alerts, configuration, encryption settings, schema, tasks, or trust store backends, the password used to generate the encryption key (which may have been obtained from an encryption settings definition) could have been inadvertently written into the backup descriptor. This problem does not affect local DB backends (like userRoot), the LDAP changelog backend, or the replication database.

  * When running certain command-line tools with an argument instructing the tool to read a password from a file, the password contained in that file could have been written into the server's tool invocation log instead of the path to that file. Affected tools include backup, create-initial-config, create-initial-proxy-config, dsreplication, enter-lockdown-mode, export-ldif, import-ldif, ldappasswordmodify, leave-lockdown-mode, manage-tasks, manage-topology, migrate-ldap-schema, parallel-update, prepare-endpoint-server, prepare-external-server, realtime-sync, rebuild-index, re-encode-entries, reload-http-connection-handler-certificates, reload-index, remove-defunct-server, restore, rotate-log, and stop-server. Other tools are not affected. Also note that this only includes passwords contained in files that were provided as command-line arguments; passwords included in the tools.properties file, or in a file referenced from tools.properties, would not have been exposed.

  In each of these cases, the files would have been written with permissions that make their contents only accessible to the system account used to run the server. Further, while administrative passwords may have been exposed in the tool invocation log, neither the passwords for regular users, nor any other data from their entries, should have been affected. We have introduced new automated tests to help ensure that such incidents do not occur in the future.

  We recommend changing any administrative passwords you fear may have been compromised as a result of this issue. If you are concerned that the passphrase for an encryption settings definition may have been exposed, then we recommend creating a new encryption settings definition that is preferred for all subsequent encryption operations, exporting your data to LDIF, and re-importing so that it will be encrypted with the new key. You also may wish to re-encrypt or destroy any existing backups, LDIF exports, or other data encrypted with a compromised key, and you may wish to sanitize or destroy any existing tool invocation log files that may contain clear-text passwords.

  - Fixed in: 7.3.0.0
  - Introduced in: 7.0.0.0
  - Support identifiers: DS-38897 DS-38908

- The following enhancements were made to the topology manager to make it easier to diagnose the connection errors:

  - Added monitoring information for all the failed outbound connections (including the time since it's been failing and the last error message seen when the failure occurred) from a server to one of its configured peers and the number of failed outbound connections.

  - Added alarms/alerts for when a server fails to connect to a peer server within a configured grace period.

  - Fixed in: 7.2.1.0
  - Introduced in: 7.0.0.0
  - Support identifiers: DS-38334 SF#00655578
- The topology manager will now raise a mirrored-subtree-manager-connection-asymmetry alarm when a server is able to establish outbound connections to its peer servers, but those peer servers are unable to establish connections back to the server within the configured grace period. The alarm is cleared when connection symmetry is achieved.

  - Fixed in: 7.2.1.0
  - Introduced in: 7.0.0.0
  - Support identifiers: DS-38344 SF#00655578
- The dsreplication tool has been fixed to work when the node being used to enable replication is currently out-of-sync with the topology master.

  - Fixed in: 7.2.1.0
  - Introduced in: 7.0.0.0
  - Support identifiers: DS-38335 SF#00655578
- Fixed two issues in which the server could have exposed some clear-text passwords in files on the server file system.

  * When creating an encrypted backup of the alarms, alerts, configuration, encryption settings, schema, tasks, or trust store backends, the password used to generate the encryption key (which may have been obtained from an encryption settings definition) could have been inadvertently written into the backup descriptor. This problem does not affect local DB backends (like userRoot), the LDAP changelog backend, or the replication database.

  * When running certain command-line tools with an argument instructing the tool to read a password from a file, the password contained in that file could have been written into the server's tool invocation log instead of the path to that file. Affected tools include backup, create-initial-config, create-initial-proxy-config, dsreplication, enter-lockdown-mode, export-ldif, import-ldif, ldappasswordmodify, leave-lockdown-mode, manage-tasks, manage-topology, migrate-ldap-schema, parallel-update, prepare-endpoint-server, prepare-external-server, realtime-sync, rebuild-index, re-encode-entries, reload-http-connection-handler-certificates, reload-index, remove-defunct-server, restore, rotate-log, and stop-server. Other tools are not affected. Also note that this only includes passwords contained in files that were provided as command-line arguments; passwords included in the tools.properties file, or in a file referenced from tools.properties, would not have been exposed.

  In each of these cases, the files would have been written with permissions that make their contents only accessible to the system account used to run the server. Further, while administrative passwords may have been exposed in the tool invocation log, neither the passwords for regular users, nor any other data from their entries, should have been affected. We have introduced new automated tests to help ensure that such incidents do not occur in the future.

  We recommend changing any administrative passwords you fear may have been compromised as a result of this issue. If you are concerned that the passphrase for an encryption settings definition may have been exposed, then we recommend creating a new encryption settings definition that is preferred for all subsequent encryption operations, exporting your data to LDIF, and re-importing so that it will be encrypted with the new key. You also may wish to re-encrypt or destroy any existing backups, LDIF

exports, or other data encrypted with a compromised key, and you may wish to sanitize or destroy any existing tool invocation log files that may contain clear-text passwords.

- Fixed in: 7.0.1.3
- Introduced in: 7.0.0.0
- Support identifiers: DS-38897 DS-38908

Upgrade Considerations

Important considerations for upgrading to this version of the Data Metrics Server

- If you have upgraded a server that is in a cluster (i.e., has a cluster name set in the Server Instance configuration object) to version 8.1, you will not be able to make cluster configuration changes until all servers with the same cluster name have been upgraded to version 8.1. If needed, you could create temporary clusters based on server versions and modify each of the servers' cluster name appropriately to minimize the impact while you are upgrading.
- Updated the "delay bind response" failure lockout action to provide an option to delay the response to bind requests initiated by non-LDAP clients (for example, when using HTTP basic authentication). This option is disabled by default because delaying the bind response for non-LDAP clients may require temporarily blocking the thread used to process the request, which could increase the risk of a denial-of-service attack. To help mitigate this risk, if you enable delayed bind responses for non LDAP clients, we recommend that you also increase the number of request handler threads for all enabled HTTP connection handlers.
- Updated setup to create a second encryption settings definition if data encryption is enabled. It will continue to create a definition for 128-bit AES encryption for use as the preferred definition to preserve backward compatibility with existing servers in the topology, but it will now also generate a definition for 256-bit AES encryption. The 256-bit AES definition may become the preferred definition in a future release, but you can use it now by first ensuring that any existing instances are updated to contain the new definition (with the "encryption-settings export" and "encryption-settings import" commands) and then making it the preferred definition (with "encryption-settings set-preferred") in all instances.

What's New

These are new features for this release of the Data Metrics Server

- In an ongoing effort to improve the use of containers for PingDirectory, several features have been implemented:

  - The --outputFile option has been added to the collect-support-data tool. You can now specify either a path, a file name, or a path and file name for the resulting CSD file. This means an administrator can run the collect-support-data tool and put the output file into a directory outside of the container, allowing access to the file without having to actually connect to the container.

  - The collect-support-data tool can now be run as a recurring task. Recurring tasks can be created using the Administration console which means that administrators do not have to connect to the container in order to run the tool.

  - A Collect Support Tool Extended Operation has been added allowing LDAP clients to initiate the collect-support-data tool and to receive the output of the request. The LDAP SDK has been updated to support this, and the --remoteServer added to the collect-support-data tool can be used to send the request to another server. In other words, you can now run collect-support-data on the command line and reference another server, possibly in a container, and retrieve the output file remotely.
- PingDirectory has a Consent REST API that allows users to create and store consents. A new feature now allows users to search for consents that have been granted to them by another party.

Known Issues/Workarounds

The following are known issues in the current version of the Data Metrics Server

- Several known issues can occur when you use the Administrative Console with Tomcat 9.0.31. You can resolve these issues by upgrading to Tomcat 9.0.33 or later.
- If you use the create-systemd-script tool to create a forking systemd service, the service is stopped by the "systemctl stop ping-directory.service" command. At that time, you can see the status using the "systemctl status ping-directory.service" command. That status might contain an indication of failure: "Active: failed (Result: exit-code)". This error has to do with the way the service exits. It is harmless.

Resolved Issues

The following issues have been resolved with this release of the Data Metrics Server:

Ticket ID	Description
DS-1046,DS-1204,DS-36547	Added support for remotely invoking the collect-support-data tool using an administrative task, and for invoking the tool on a regular basis as a recurring task. The tool has also been updated to add an outputPath argument to allow specifying the path or name to use for the output file.
DS-37829	The "create-systemd-script" CLI now creates a "forking" service file since Ping services are started by a process (the "start-server" script) that is different than the actual service process.
DS-38122	Added support for an extended operation that can be used to invoke the collect-support-data tool from a remote system and stream the output and resulting support data archive back to the client. The collect-support-data command-line tool has been updated to support this capability through the new --useRemoteServer argument.
DS-38535	Fixed an issue that could cause the server to generate an administrative alert about an uncaught exception when trying to send data on a TLS-encrypted connection that is no longer valid.
DS-39798	Fixed a bug in which SEMI_AGGRESSIVE and AGGRESSIVE JVM Tuning Parameters were previously allowed to both be selected.
DS-40356	Updated the manage-profile tool to prevent displaying warnings about offline config changes when starting the server.

Ticket ID	Description
DS-40532	Added a logging-error-behavior property to the log publisher, periodic stats logger plugin, and monitor history plugin configuration that can be used to specify the behavior the server should exhibit if an error occurs while attempting logging-related processing. By default, the server will preserve its previous behavior of writing a message to standard error, but it can be configured to enter lockdown mode on a logging error, in which the server will report itself as unavailable and will only accept requests from accounts with the lockdown-mode privilege and only from clients communicating over a loopback interface.
DS-40551	Fixed an issue that could prevent some tools from running properly with an encrypted tools.properties file.
DS-40567	A license is now always required when using the manage-profile replace-profile tool.
DS-40746	Updated the logic that the server uses to select an appropriate default set of TLS cipher suites.
DS-40806	Fixed an issue that could cause the shutdown process to stall if the server is configured to use TCP to communicate with a StatsD endpoint that has become unresponsive.
DS-40889	Fixed an issue with recurring exec tasks where the working-directory attribute was ignored.
DS-41074	Fixed an issue with the way the server reports memory usage after completing an explicitly requested garbage collection.
DS-41086	Updated the StatsD monitoring endpoint to replace any spaces, commas, or colons with underscores, and remove and single quotes or double quotes in sent metric lines. This simplifies parsing of the produced metrics.
DS-41118	A gauge called HTTP Processing (Percent) is now available. This gauge measures the server's capacity to process new incoming HTTP requests.
DS-41126	Updated the server to make the general monitor entry available to JMX clients.

Ticket ID	Description
DS-41142	Improved debugging support for Server SDK extensions. If debugging is enabled, the server will now generate a debug message whenever it invokes an extension. For some extension methods that return a value, the server will also generate a debug message with that return value.
DS-41206	Fixed a memory leak when performing SCIM queries on the Directory Server.
DS-41235	Updated the cn=Cluster subtree to prevent clustered configuration changes when servers in the cluster have mixed versions. To make clustered configuration changes, either update all servers in the cluster to the same version, or temporarily create separate clusters by server version by changing the cluster-name property on the server instance configuration objects.
DS-41236	To avoid inconsistencies, changing clustered configuration will now require all servers in the cluster to be on the same product version. Servers will not pull any clustered configuration from the master of the cluster if they are on a different product version.
DS-41261	Fixed an issue with manage-profile replace-profile where certain configuration changes for recurring task chains were not being applied.
DS-41289	Fixed an issue that prevented password changes for topology administrators unless their password policy was configured to allow pre-encoded passwords.
DS-41301	Addressed an issue that could lead to slow, off-heap memory growth. This only occurred on servers whose cn=Version,cn=monitor entry was retrieved frequently.
DS-41333	Added an ssl-client-auth-policy configuration property to the HTTP connection handler to provide support for mutual TLS authentication.
DS-41366	Updated the base monitor entry to include locationName and locationDN attributes if the server is configured with a location.
DS-41396	Updated the Server SDK to add ClientContext and OperationContext methods for obtaining the name and DN of the associated client connection policy.

Ticket ID	Description
DS-41400	Updated the file servlet HTTP servlet extension to add support for requiring authentication in order to access the content. Access may optionally be limited to members of a specified set of groups.
DS-41731	Fixed an issue that could prevent setup from generating a self-signed certificate for systems with non-ASCII hostnames.
DS-41762	Fixed an issue where mirrored subtree polling could produce config archive files that were identical or ignored the configured insignificant attributes list.
DS-41818	Added the --zip argument to the manage-profile generate-profile subcommand, which can be used to generate a zipped server profile.
DS-41820	Added an administrative task that may be used to generate a server profile and a corresponding recurring task that may be used to invoke the task on a regular basis.
DS-41821	Added an instance root file servlet to the default configuration. HTTPS requests to /instance-root by authenticated users with the file-servlet-access privilege will be granted access to files within the server instance root.
DS-41850	Servers running on Linux will now log a warning about possible performance impacts if the current memory control group has memory.swappiness set to a nonzero value.
DS-42006	The server now warns the administrator at startup if there are multiple versions of the same jar listed in the classpath, and the first one in the classpath is not the newest one.
DS-42033	Addressed an issue where some tools would throw a NullPointerException if a server was configured with a custom global result code map.
DS-42387	Updated the **`manage-profile generate-profile`** subcommand to exclude files in the `ldif/` and `bak/` directories by default when generating a server profile. If necessary, you can manually include those directories using the `--includePath` argument.

PingDataMetrics 8.0.0.5 release notes

Critical fixes

This release of the Data Metrics Server addresses critical issues from earlier versions. Update all affected servers appropriately.

No critical issues have been identified.

Ping Data Metrics Server 8.0.0.3 release notes

Critical Fixes

This release of the Data Metrics Server addresses critical issues from earlier versions. Update all affected servers appropriately.

No critical issues have been identified

Resolved Issues

The following issues have been resolved with this release of the Data Metrics Server:

Ticket ID	Description
DS-38535	Fixed an issue that could cause the server to generate an administrative alert about an uncaught exception when trying to send data on a TLS-encrypted connection that is no longer valid.

Ticket ID	Description
DS-43288	Updated setup and the replace-certificate tool to improve the way we generate self-signed certificates and certificate signing requests to make them more palatable to clients. To reduce the frequency with which administrators had to replace self-signed certificates, we previously used a very long lifetime for self-signed certificates generated by setup or the replace-certificate tool. However, some clients (especially web browsers and other HTTP clients) have started more strenuously objecting to certificates to long lifetimes, so we now generate self-signed certificates with a one-year validity period. The inter-server certificate (which is used internally within the server and does not get exposed to normal clients) is still created with a twenty-year lifetime. Also, the replace-certificate tool's interactive mode has been updated to improve the process that it uses to obtain information to include in the subject DN and subject alternative name extension for self-signed certificates and certificate signing requests. The following changes have been made in accordance with CA/Browser Forum guidelines: * When selecting the subject DN for the certificate, we listed a number of common attributes that may be used, including CN, OU, O, L, ST, and C. We previously indicated that CN attribute was recommended. We now also indicate that the O and C attributes are recommended as well. * When obtaining the list of DNS names to include in the subject alternative name extension, we previously suggested all names that we could find associated with interfaces on the local system. In many cases, we now omit non-qualified names and names that are associated with loopback interfaces. We will also warn about any attempts to add unqualified or invalid names to the list. * When obtaining the list of IP addresses to include in the subject alternative name extension, we previously suggested all addresses associated with all network interfaces on the system. We no longer suggest any IP addresses associated with loopback interfaces, and we no longer suggest any IP addresses associated in IANA-reserved ranges (for example, addresses reserved for private-use networks). The tool will now warn about attempts to add these addresses for inclusion in the subject alternative name extension.

Ticket ID	Description
DS-43480	Updated the system information monitor provider to restrict the set of environment variables that may be included. Previously, the monitor entry included information about all defined environment variables, as that information can be useful for diagnostic purposes. However, some deployments may include credentials, secret keys, or other sensitive information in environment variables, and that should not be exposed in the monitor. The server will now only include values from a predefined set of environment variables that are expected to be the most useful for troubleshooting problems, and that are not expected to contain sensitive information.
DS-43632	Fixed an issue where the "format" field is omitted from the list of operational attribute schemas in the Directory REST API.
DS-43651	The Security Guide is now available online at pingidentity.com and has been removed from the server packaging.

Ping Data Metrics Server 8.0.0.2 Release Notes

Critical Fixes

This release of the Ping Data Metrics Server addresses critical issues from earlier versions. Update all affected servers appropriately.

No critical issues have been identified

Resolved Issues

The following issues have been resolved with this release of the Ping Data Metrics Server :

Ticket ID	Description
DS-40551	Fixed an issue that could prevent some tools from running properly with an encrypted tools.properties file.
DS-40828	Fixed an issue where some state associated with a JMX connection was not freed after the connection was closed. This led to a slow memory leak in servers that were monitored by an application that created a new JMX connection each polling interval.

Ticket ID	Description
DS-41074	Fixed an issue with the way the server reports memory usage after completing an explicitly requested garbage collection.
DS-41126	Updated the server to make the general monitor entry available to JMX clients.
DS-41235	Updated the cn=Cluster subtree to prevent clustered configuration changes when servers in the cluster have mixed versions. To make clustered configuration changes, either update all servers in the cluster to the same version, or temporarily create separate clusters by server version by changing the cluster-name property on the server instance configuration objects.
DS-41236	To avoid inconsistencies, changing clustered configuration will now require all servers in the cluster to be on the same product version. Servers will not pull any clustered configuration from the master of the cluster if they are on a different product version.
DS-41261	Fixed an issue with manage-profile replace-profile where certain configuration changes for recurring task chains were not being applied.
DS-41289	Fixed an issue that prevented password changes for topology administrators unless their password policy was configured to allow pre-encoded passwords.
DS-42609	Fixed an issue in which the Directory REST API could fail to decode certain credentials when using basic authentication.
DS-42812	Upgrade to jetty 9.4.30

PingDataMetrics Server 8.0.0.1 Release Notes

Critical Fixes

This release of the Data Metrics Server addresses critical issues from earlier versions. Update all affected servers appropriately.

- Fixed a memory leak when performing SCIM queries on .
  - Fixed in: 8.0.0.1

    Introduced in: 7.2.0.0

    Support identifiers: DS-41206 SF#00681395
- Addressed an issue that could lead to slow off-heap memory growth. This only occurred on servers whose cn=Version,cn=monitor entry was retrieved frequently.
  - Fixed in: 8.0.0.1

    Introduced in: 5.2.0.0

    Support identifiers: DS-41301
- The following enhancements were made to the topology manager to make it easier to diagnose connection errors:

  - Added monitoring information for all the failed outbound connections (including the time since it's been failing and the last error message seen when the failure occurred) from a server to one of its configured peers and the number of failed outbound connections.

  - Added alarms/alerts for when a server fails to connect to a peer server within a configured grace period.
  - Fixed in: 7.3.0.0
  - Introduced in: 7.0.0.0
  - Support identifiers: DS-38334 SF#00655578
- The topology manager will now raise a mirrored-subtree-manager-connection-asymmetry alarm when a server is able to establish outbound connections to its peer servers, but those peer servers are unable to establish connections back to the server within the configured grace period. The alarm is cleared as soon as there is connection symmetry.
  - Fixed in: 7.3.0.0
  - Introduced in: 7.0.0.0
  - Support identifiers: DS-38344 SF#00655578
- The dsreplication tool has been fixed to work when the node being used to enable replication is currently out-of-sync with the topology master.
  - Fixed in: 7.3.0.0
  - Introduced in: 7.0.0.0
  - Support identifiers: DS-38335 SF#00655578
- Fixed two issues in which the server could have exposed some clear-text passwords in files on the server file system.

  * When creating an encrypted backup of the alarms, alerts, configuration, encryption settings, schema, tasks, or trust store backends, the password used to generate the encryption key (which may have been obtained from an encryption settings definition) could have been inadvertently written into the backup descriptor. This problem does not affect local DB backends (like userRoot), the LDAP changelog backend, or the replication database.

  * When running certain command-line tools with an argument instructing the tool to read a password from a file, the password contained in that file could have been written into the server's tool invocation log instead of the path to that file. Affected tools include backup, create-initial-config, create-initial-proxy-config, dsreplication, enter-lockdown-mode, export-ldif, import-ldif, ldappasswordmodify, leave-lockdown-mode, manage-tasks, manage-topology, migrate-ldap-schema, parallel-update, prepare-endpoint-server, prepare-external-server, realtime-sync, rebuild-index, re-encode-entries, reload-http-connection-handler-certificates, reload-index, remove-defunct-server, restore, rotate-log, and stop-server. Other tools are not affected. Also note that this only includes passwords contained in files that

were provided as command-line arguments; passwords included in the tools.properties file, or in a file referenced from tools.properties, would not have been exposed.

In each of these cases, the files would have been written with permissions that make their contents only accessible to the system account used to run the server. Further, while administrative passwords may have been exposed in the tool invocation log, neither the passwords for regular users, nor any other data from their entries, should have been affected. We have introduced new automated tests to help ensure that such incidents do not occur in the future.

We recommend changing any administrative passwords you fear may have been compromised as a result of this issue. If you are concerned that the passphrase for an encryption settings definition may have been exposed, then we recommend creating a new encryption settings definition that is preferred for all subsequent encryption operations, exporting your data to LDIF, and re-importing so that it will be encrypted with the new key. You also may wish to re-encrypt or destroy any existing backups, LDIF exports, or other data encrypted with a compromised key, and you may wish to sanitize or destroy any existing tool invocation log files that may contain clear-text passwords.

  - Fixed in: 7.3.0.0
  - Introduced in: 7.0.0.0
  - Support identifiers: DS-38897 DS-38908
- The following enhancements were made to the topology manager to make it easier to diagnose the connection errors:

  - Added monitoring information for all the failed outbound connections (including the time since it's been failing and the last error message seen when the failure occurred) from a server to one of its configured peers and the number of failed outbound connections.

  - Added alarms/alerts for when a server fails to connect to a peer server within a configured grace period.

  - Fixed in: 7.2.1.0
  - Introduced in: 7.0.0.0
  - Support identifiers: DS-38334 SF#00655578
- The topology manager will now raise a mirrored-subtree-manager-connection-asymmetry alarm when a server is able to establish outbound connections to its peer servers, but those peer servers are unable to establish connections back to the server within the configured grace period. The alarm is cleared when connection symmetry is achieved.

  - Fixed in: 7.2.1.0
  - Introduced in: 7.0.0.0
  - Support identifiers: DS-38344 SF#00655578
- The dsreplication tool has been fixed to work when the node being used to enable replication is currently out-of-sync with the topology master.

  - Fixed in: 7.2.1.0
  - Introduced in: 7.0.0.0
  - Support identifiers: DS-38335 SF#00655578
- Fixed two issues in which the server could have exposed some clear-text passwords in files on the server file system.

  * When creating an encrypted backup of the alarms, alerts, configuration, encryption settings, schema, tasks, or trust store backends, the password used to generate the encryption key (which may have been obtained from an encryption settings definition) could have been inadvertently written into the backup descriptor. This problem does not affect local DB backends (like userRoot), the LDAP changelog backend, or the replication database.

  * When running certain command-line tools with an argument instructing the tool to read a password from a file, the password contained in that file could have been written into the server's tool invocation log instead of the path to that file. Affected tools include backup, create-initial-config, create-initial-proxy-config, dsreplication, enter-lockdown-mode, export-ldif, import-ldif, ldappasswordmodify, leave-

lockdown-mode, manage-tasks, manage-topology, migrate-ldap-schema, parallel-update, prepare-endpoint-server, prepare-external-server, realtime-sync, rebuild-index, re-encode-entries, reload-http-connection-handler-certificates, reload-index, remove-defunct-server, restore, rotate-log, and stop-server. Other tools are not affected. Also note that this only includes passwords contained in files that were provided as command-line arguments; passwords included in the tools.properties file, or in a file referenced from tools.properties, would not have been exposed.

In each of these cases, the files would have been written with permissions that make their contents only accessible to the system account used to run the server. Further, while administrative passwords may have been exposed in the tool invocation log, neither the passwords for regular users, nor any other data from their entries, should have been affected. We have introduced new automated tests to help ensure that such incidents do not occur in the future.

We recommend changing any administrative passwords you fear may have been compromised as a result of this issue. If you are concerned that the passphrase for an encryption settings definition may have been exposed, then we recommend creating a new encryption settings definition that is preferred for all subsequent encryption operations, exporting your data to LDIF, and re-importing so that it will be encrypted with the new key. You also may wish to re-encrypt or destroy any existing backups, LDIF exports, or other data encrypted with a compromised key, and you may wish to sanitize or destroy any existing tool invocation log files that may contain clear-text passwords.

  - Fixed in: 7.0.1.3
  - Introduced in: 7.0.0.0
  - Support identifiers: DS-38897 DS-38908
- The server can now detect an "out of file handles" situation on the operating system, and shut down to prevent running in an unreliable state.
  - Fixed in: 5.1.0.0
  - Introduced in: 2.1.0.0
  - Support identifiers: DS-12579 SF#2655
- Disabled support for SSLv3 by default in the LDAP, HTTP, and JMX connection handlers, and for replication communication. The recently-discovered POODLE vulnerability could potentially allow a network attacker to determine the plaintext behind an SSLv3-encrypted session, which would effectively negate the primary benefit of the encryption.

  SSLv3 was initially defined in 1996, but was supplanted by the release of the TLSv1 definition in 1999 (and subsequently by TLSv1.1 in 2006 and TLSv1.2 in 2008). These newer TLS protocols are not susceptible to the POODLE vulnerability, and the server has supported them (and preferred them over SSLv3) for many years. The act of disabling SSLv3 by default should not have any adverse effect on clients that support any of the newer TLS protocols. However, if there are any legacy client applications that attempt to communicate securely but do not support the newer TLS protocols, they should be updated to support the newer protocols. In the event that there are known clients that do not support any security protocol newer than SSLv3 and that cannot be immediately updated to support a newer protocol, SSLv3 support can be re-enabled using the newly-introduced allowed-insecure-tls-protocol global configuration property. However, since communication using SSLv3 can no longer be considered secure, it is strongly recommended that every effort be made to update all known clients still using SSLv3.

  It is possible to use the server access log to identify LDAP clients that use SSLv3 to communicate with the server. Whenever an LDAP client establishes a secure connection to the server, or whenever a client uses the StartTLS extended operation to secure an existing plaintext connection, the server will generate a SECURITY-NEGOTIATION access log message. The "protocol" element of a SECURITY-NEGOTIATION access log message specifies the name of the security protocol that has been negotiated between the client and the server, and any SECURITY-NEGOTIATION messages with a protocol of "SSLv3" suggest that the associated client is vulnerable to the POODLE attack. In addition,

if any connections are terminated for attempting to use the disallowed SSLv3 protocol, the access log message for that disconnect should include a message stating the reason for the termination.

- Fixed in: 5.0.0.0
- Introduced in: 2.1.0.0
- Support identifiers: DS-11782

Resolved Issues

The following issues have been resolved with this release of the Data Metrics Server:

Ticket ID	Description
DS-40532	Added a logging-error-behavior property to the log publisher, periodic stats logger plugin, and monitor history plugin configuration that can be used to specify the behavior the server should exhibit if an error occurs while attempting logging-related processing. By default, the server will preserve its previous behavior of writing a message to standard error, but it can be configured to enter lockdown mode on a logging error, in which the server will report itself as unavailable and will only accept requests from accounts with the lockdown-mode privilege and only from clients communicating over a loopback interface.
DS-41206	Fixed a memory leak when performing SCIM queries on the Directory Server.

PingDataMetrics Server 8.0.0.0 Release Notes

Critical Fixes

This release of PingDataMetrics Server addresses critical issues from earlier versions. Update all affected servers appropriately.

No critical issues have been identified

Known Issues and Workarounds

The following are known issues in the current version of PingDataMetrics Server:

- The following are suggested solutions for problems with slow DNS:
  - Maintain a connection pool in the client app rather than opening new connections for each bind.
  - Add appropriate records, including PTR records, to DNS.
  - Add `options timeout:1` in the `/etc/resolv.conf` file and/or `options single-request`.
  - If IPv6 requests specifically are causing issues, add `-Djava.net.preferIPv4Stack=true` to the `start-server.java-args` line in PingDirectory's `config/java.properties` file, run `bin/dsjavaproperties`, and restart the server to stop the issuance of IPv6 PTR requests.
- Some server tools, such as `dsreplication`, `collect-support-data`, and `rebuild-index`, will fail with errors if they are run with an encrypted `tools.properties` file.
  - **Workaround:** Add the `--noPropertiesFile` argument to the server tools to prevent them from pulling information from the encrypted file.
- The working directory value used by exec tasks is not implemented for recurring exec tasks.

Resolved Issues

The following issues have been resolved with this release of PingDataMetrics Server:

Ticket ID	Description
DS-17278	Added a `cn=Server Status Timeline,cn=monitor` monitor entry to track a history of the local server's last 100 status changes and their timestamps. Updated the LDAP external server monitor to include attributes tracking health check state changes for external servers. The new attributes include the number of times a health check transition has occurred, timestamps of the most recent transitions, and messages associated with the most recent transitions.
DS-37881	The PingFederate Access Token Validator will now refresh its cached value of the PingFederate server's token introspection endpoint. A new attribute, `endpoint-cache-refresh`, has been added to the PingFederate Access Token Validator, which will determine how often this refresh occurs.
DS-37955	To support multiple trace loggers, each trace logger now has its own resource key, which is shown in the `Resource` column in the output of `status`. This key allows multiple alarms, due to sensitive message types for multiple trace loggers.
DS-38053	The JWT Access Token Validator no longer requires a restart after a change to one of its signing certificates.
DS-38832	Fixed an issue that could cause the server to leak a small amount of memory each time it failed to establish an LDAP connection to another server.
DS-38867	Updated the PBKDF2 password storage scheme to add support for variants that use the 256-bit, 384-bit, and 512-bit SHA-2 digest algorithms. At present, the SHA-1 variant remains the default to preserve backward compatibility with older versions. Also, in accordance with the recommendations in NIST SP 800-63B, we have increased the default iteration count from 4096 to 10,000, and the default salt length from 64 bits to 128 bits.
DS-38869	Updated the `remove-defunct-server` tool's `--ignoreOnline` option. When using `--ignoreOnline` in a mixed-version environment, all servers must support the option.
DS-39176, DS-39308	Updated the Groovy scripting language version to 2.5.7. For a list of changes, visit *groovy-lang.org* and view the Groovy 2.5 release notes. As of this release, only the core Groovy runtime and the groovy-json module are bundled with the server. To deploy a Groovy-scripted Server SDK extension that requires a Groovy module not bundled with the server, such as groovy-xml or groovy-sql, download the appropriate jar file from *groovy-lang.org* and place it in the server's `lib/extensions` directory.
DS-39253	Added a `replace-certificate` tool, which can help an administrator replace the listener or inter-server certificate for a server instance.

Ticket ID	Description
DS-39325	Removed the legacy product-specific scripts for starting and stopping the server. These include: ▪ `start-ds` and `stop-ds` for Directory Server ▪ `start-proxy` and `stop-proxy` for Directory Proxy Server ▪ `start-sync` and `stop-sync` for Data Sync Server ▪ `start-metrics-engine` and `stop-metrics-engine` for These legacy scripts had been deprecated for several releases in favor of the more general start-server and stop-server scripts, and they displayed a warning message about their upcoming removal if they were invoked. If you still have dependencies on these legacy product-specific scripts, you will need to update them to reference the general start-server and stop-server scripts instead. If it is not feasible to update these references immediately, you may create symbolic links that use the legacy script names and point at the start-server and stop-server scripts.
DS-39373	Preserve the privileges that are explicitly set on the admin user when migrating from the admin backend to the topology registry.
DS-39518	Fixed an issue in which escaped characters in schema extensions may not be handled properly. If used in attribute type constraints (such as `X-VALUE-REGEX`), this could cause unexpected or incorrect behavior.
DS-39592	HTTP External Servers have a new attribute, `ssl-cert-nickname`, which defines the alias of a specific certificate within their keystore to be used as a client certificate.
DS-39626, DS-40357	The trace log publisher will now record an access token's scopes after the token is successfully validated.
DS-39654	Added support for the `--topologyFilePath` argument to the `manage-topology add-server` subcommand.
DS-39715	Updated the Server SDK to add support for sending email messages.
DS-39857	Added the StatsD monitoring endpoint. When the Stats Collector Plugin is enabled, this endpoint sends metric data from the server in StatsD format to the configured destination.
DS-39908	Added a new JVM-default trust manager provider that can be used to automatically trust any certificate signed by an authority included in the JVM's default set of trusted issuers. Also, updated other trust manager providers to offer an option to use the JVM-default trust addition to the trust that they normally provide.
DS-40114	Added a new `cn=Status Health Summary,cn=monitor` monitor entry that provides a summary of the server's current assessment of its health. This simplifies monitoring with third party tools that support retrieving monitoring data over JMX. The Periodic Stats Logger has also been updated to allow some of this monitoring information to be logged. No new information is logged by default.
DS-40354	Fixed a problem with `config-diff` when writing properties that span multiple lines using the `--prettyPrint` argument.

Ticket ID	Description
DS-40366	Fixed an issue where the server was attempting to connect by an IP address rather than a hostname when DNS lookup was successful.
DS-40377	Added support for logging to a JSON file in the Periodic Stats Logger Plugin.
DS-40517	Added metrics for status summary, replication database, and LDAP changelog to the Stats Collector Plugin.
DS-40556	Added support for specifying a working directory for exec tasks.
DS-40730	Updated the `encrypt-file` tool to prevent using the same path for both the input file and the output file.
DS-40771	Added a `--duration` argument to `collect-support-data`. When used, only the log files covering the specified duration before the current time will be collected.

Delegated Admin Release Notes

Delegated Admin 4.4.0 - December 2020

The following enhancements and resolved issues are included in this release.

Enhancements

These are new features for this release of the Delegated Administration application:

- Delegated administrative users can now create a user account without setting a password if the password is not required by the server. This is only true for user creation. Resetting a password still requires a valid value for the password.
- Previously, users could download a full report of users and group membership. Now the administrative user can preview the report and apply filters to limit either the entries themselves or to filter which columns are displayed. The user can also sort the report by columns. The user can then download the resulting report with all the filters applied.
- The Delegated Administrator user interface now allows you to move and display the password field anywhere on the **Create User** form. Specify this in the configuration of the REST Resource Type with the Password Attribute Category and Password Display Order Index properties. If you do not specify a category, the password field appears at the top of the page.
- The Delegated Administration application does not require delegated administrative users to understand the underlying technology supporting the account management features, such as LDAP. PingDirectory is an LDAP server and, therefore, uses Distinguished Names (DNs) to identify entries in the directory. The Delegated Administration application allows the user to select a user entry based on its DN without displaying the actual value of the DN.
- Account management improvements allow administrators to view information about account and password state, such as password expiration date, account lockout status, and last login date and time.
- Administrative users can now bulk upload user entries from a CSV file. The users need the proper rights for the resource types on which they wish to create user entries. The information in the file to upload must be based on the attributes configured for the resource type. The upload creates user entries if they do not exist, and optionally adds the entries to a group. If a user entry exists, the upload updates the entry. You can also upload generic entries.
- Previously, administrative users could create, preview, and download reports of users, group membership, or other generic resources to a CSV file. With this release, you can choose to not present the download and/or upload options to certain administrative users.

- Completing the basic entry management functionality, administrative users with the proper rights for the resource type can now delete user, group, and generic entries.

Upgrade considerations

Upgrade considerations are no longer part of the release notes. That information is now in *Upgrade considerations* on page 1244.

Resolved issues

The following table identifies issues that have been resolved with this release of Delegated Admin.

Ticket ID	Description
DS-42793	Fixed an issue where an infinite spinner would intermittently appear when signing on.
DS-41913	Added the ability to move the Password field on the **Create User** form.

Known issues and limitations

- Delegated Admin 4.2.0 has an intermittent, infinite spinner issue when signing on. To address the issue and continue using PingDirectory 8.1.0.0, install Delegated Admin 4.2.1. Alternatively, you can upgrade to Delegated Admin 4.4.0 to address the issue; however, that upgrade requires upgrading to PingDirectory 8.2.0.0.
- For versions 4.2.0-4.4.0, Delegated Admin produces an error on edit or create of resource types that contain an attribute with an unrecognized data type. For supported data types, see *Configuring attributes and attribute search on PingDirectory Server* on page 1254. Delegated Admin 4.4.1 addresses this error.

Previous releases

Release Notes for earlier versions of Delegated Admin are included for reference.

Delegated Admin 4.2.0 Release Notes

Upgrade Considerations

If you are running Delegated Administration application version 3.5 or older, you will need to upgrade it to the latest version if you want to use PingDirectory 8.0 or newer.

What's New

These are new features for this release of PingDirectory Delegated Admin:

- Delegated Admin 4.1
  - For Delegated Admin, administrators can be assigned rights to manage different types of resources such as users and groups. Previously, administrators who can only manage users in specified groups could not create users to put them in the one of the groups they managed. Now, administrators with the ability to manage and create groups can create users within the group(s) they manage.
  - REST resource types define the types of objects that can be managed by delegated administrators. Typical REST resource types include users and groups which can be configured based on the sub-tree and a predefined search filter. A new include-filter property has been added to REST resource types to differentiate different resource types that are within the same subtree and have the same objectclass.

- Delegated Admin 4.2
  - Administrative users can now create and download reports of users, group membership or other generic resource to a CSV file. The users need the proper rights for the resource types on which they wish to report. The information stored in the downloaded file will be based on the attributes configured for the resource type. For group resource types, each report generated will be for a single group.
  - Added a new reference permission to Delegated Admin resource rights configuration which allows the administrator to reference resources when selecting a parent for a new resource. This permission differs from read permission in that the app will not show an option to manage the parent resource type in the Delegated Administrator app.
  - In many circumstances, LDAP attributes within an entry contain multiple values. The application can now display multi-valued attributes which are not handled by custom UI form fields.
  - A privileged administrator for a hosting company can onboard a new tenant administrator to manage resources for the tenant's own organization without requiring additional configuration from the Administrative Console or command line.

Known Issues/Workarounds

The following are known issues in the current version of PingDirectory Delegated Admin:

- PingDirectory Delegated Admin does not always show the account locked status correctly if the server password policy is configured to require the user to reset their password after the password has been administratively reset. This can be corrected by enabling the Password Policy State JSON virtual attribute for the user object class. For example:

```
dsconfig set-virtual-attribute-prop --name "Password Policy State JSON" \
--set enabled:true --set require-explicit-request-by-name:true \
--set "filter:(objectClass=person)"
```

Resolved Issues

The following table identifies issues that have been resolved with this release of Delegated Admin:

Ticket ID	Description
DS-41819	Fixed an issue where a Delegated Administrator was unable to add or remove themselves to or from a group to which they had rights to manage group membership.
DS-41848	Fixed an issue where edits modifying only the case or capitalization of an attribute value did not take effect.
DS-42387	Updated the **`manage-profile generate-profile`** subcommand to exclude files in the `ldif/` and `bak/` directories by default when generating a server profile. If necessary, you can manually include those directories using the `--includePath` argument.

Delegated Admin 4.1.0 Release Notes

Upgrade Considerations

Upgrade directly to this release from versions prior to 7.0 is not supported. As a workaround, upgrade from a pre-7.0 version to 8.0, and then upgrade to this version.

What's New

These are new features for this release of :

- Administrators can be assigned rights to manage different types of resources such as users and groups. Previously, administrators who can only manage users in specified groups could not create users to put them in the one of the groups they managed. Now, administrators with the ability to manage and create groups can create users within the group(s) they manage.
- When the Delegated admin rights for a User REST resource type have admin scope resources-in-specific-groups, a user is added to one of the configured groups when the user is created.
- An example file for JSON attributes, example-json.html, has been added and can be located alongside other existing example files for custom UI form fields. This example file supports a single value and uses the expected JSON structure of ubidEmailJSON, an available JSON attribute type for Ping Directory. Multiple values are also supported, please see the example-multivalue.html for an example of how to handle that use-case. As with any custom UI form field, modifications will likely be necessary and some technical knowledge will be required.
- Multivalue attributes may now be handled as part of custom UI form fields. Contrary to single value attributes, an array of values will need to be sent up to the form when building out the custom field. An example file, example-multivalue.html, has been provided to demonstrate a string-based, multivalue attribute as a custom UI form field and can be located alongside other existing example files. As with any custom UI form field, modifications will likely be necessary and some technical knowledge will be required.

Known Issues/Workarounds

The following are known issues in this version of Delegated Admin:

- In environments with Proxy and multiple backend servers, there is a known issue using the fewest-operation load-balancing algorithm with the Delegated Admin app. Constructed attributes sometimes do not get updated when the attributes they reference are edited. Using a failover load-balancing algorithm is a workaround for this issue. (DS-40977)

Resolved Issues

The following table identifies issues that have been resolved with this release of Delegated Admin:

Ticket ID	Description
DS-40938	Fixed an issue when Delegated Admin is installed on the Directory Proxy Server where a constructed attribute that is set to be read-only may not update in the app immediately after one of the attributes it references is updated.
DS-41066	Fixed an issue where JSON object values for attributes caused an application error when attempting to display them in the UI.

Ticket ID	Description
DS-41275	It is now a configuration error to enable two REST resource types with the same endpoint (ignoring differences in case) at the same time. It is allowed to have two resource types with the same endpoint if one or both of them are not enabled.
DS-41669	An example file for JSON attributes, example-json.html, has been added and can be located alongside other existing example files for custom UI form fields. This example file supports a single value and uses the expected JSON structure of ubidEmailJSON, an available JSON attribute type for Ping Directory. Multiple values are also supported, please see the example-multivalue.html for an example of how to handle that use-case. As with any custom UI form field, modifications will likely be necessary and some technical knowledge will be required.
DS-41715	Multivalue attributes may now be handled as part of custom UI form fields. Contrary to single value attributes, an array of values will need to be sent up to the form when building out the custom field. An example file, example-multivalue.html, has been provided to demonstrate a string-based, multivalue attribute as a custom UI form field and can be located alongside other existing example files. As with any custom UI form field, modifications will likely be necessary and some technical knowledge will be required.

Delegated Admin 4.0.0 Release Notes

Upgrade Considerations

Important considerations for upgrading to this version of Delegated Admin:

PingDirectory 8.0.0.0 is now the minimum required version to use with Delegated Admin 4.0.0. If you are on an older version of PingDirectory, it will be necessary to upgrade to PingDirectory 8.0.0.0 to maintain compatibility.

Delegated Admin 3.5.1 is compatible with PingDirectory Server 8.0.0.0. However, versions of Delegated Admin that are earlier than 4.0.0 will not be compatible with versions of PingDirectory Server that are later than 8.0.0.0.

By default, the display name for the logged in delegated admin will no longer display. To re-configure this functionality, please see the section on enabling the `name` attribute to be returned with the OIDC id token. See the Delegated Admin Guide for more information.

The `delegated-admin-template.dsconfig` file has been updated to allow for `generate-password` extended requests and password validation details request controls. This change is not applied during an update. You must run the following two **`dsconfig`** commands when updating :

```
dsconfig set-access-control-handler-prop --add \
'global-aci:(extop="1.3.6.1.4.1.30221.2.6.62")(version 3.0; \
acl "Authenticated access to the generate-password extended \
```

```
request for the Delegated Admin API"; allow (read) userdn="ldap:///all";)'
```

```
dsconfig set-access-control-handler-prop \
--add 'global-aci:(targetcontrol="1.3.6.1.4.1.30221.2.5.40")\
(version 3.0;acl "Authenticated access to the password validation details
 request \
control for the Delegated Admin API"; allow (read) userdn="ldap:///all";)'
```

What's New

These are new features for this release of :

- Invite new users via e-mail: Taking advantage of the new e-mail notification capabilities of 8.0.0.0, now administrators can configure the service so that when a delegated admin creates a new user, the server can send an HTML e-mail to tell the new user their password and invite them to use their new account. Combine this with PingFederate self-service profile management to invite the new user to complete their profile.
- When creating users or resetting passwords, delegated admins now have the option to type in the new password or have the server generate a password. Previously the application only supported server-generated passwords.
- More flexibility in delegating the management of user profiles: Now administrators can configure the service so that delegated admins of one type, such as Employee, can create and manage users of other types, such as Customers or Members. Previously delegated admins could only create and manage users of their own type.

Known Issues/Workarounds

The following are known issues in this version of Delegated Admin:

-
- Deploying the Admin Console to an external container using JDK 11 requires downloading the following dependencies and making them available at runtime (for example, by copying them to the `WEB-INF/lib` directory of the exploded WAR file).
  - groupId:jakarta.xml.bind, artifactId:jakarta.xml.bind-api, version:2.3.2
  - groupId:org.glassfish.jaxb, artifactId:jaxb-runtime, version:2.3.2

  Workaround: Deploy the Console in an external container using JDK 8.

Resolved Issues

The following table identifies issues that have been resolved with this release of Delegated Admin:

Ticket ID	Description
DS-39352	When adding a user to groups and there were no non-member groups to display, the notice text did not use proper context to reflect this state.
DS-39782	Constructed attributes were not updated in the application after their associated attributes were edited. A page refresh or subsequent data request was required to reflect the change for the constructed attribute in the application.

Ticket ID	Description
DS-40690	This scenario occurred when a delegated admin only had permission to edit users in certain groups. If the delegated admin then went to a user's profile and removed them from the group which governed permission over that user, this action resulted in an application error.
DS-41715	Multivalue attributes may now be handled as part of custom UI form fields. Contrary to single value attributes, an array of values will need to be sent up to the form when building out the custom field. An example file, example-multivalue.html, has been provided to demonstrate a string-based, multivalue attribute as a custom UI form field and can be located alongside other existing example files. As with any custom UI form field, modifications will likely be necessary and some technical knowledge will be required.

Delegated Admin 3.5.1 Release Notes

Resolved Issues

The following table identifies issues that have been resolved with this release of Delegated Admin.

Ticket ID	Description
DS-38745, DS-39639	Previously, when port 443 was used but not specified in the PingFederate Base URL, the system returned an `Invalid issuer in token` message. With this fix, the value for **`window.PF_PORT`** can be left blank in `config.js` to match the expected Base URL that the PingFederate token supplies.

Delegated Admin 3.5.0 Release Notes

Upgrade Considerations

Important considerations for upgrading to this version of Delegated Admin:

The file `remove-sample-directory-data-aci.ldif`, which is provided with the Delegated Admin installation package, was updated to reinstate permissions for users to change their own passwords (self-service password reset). If you have used an earlier version of this LDIF file, consider manually adding the following ACI from the updated version:

```
dn: dc=example,dc=com
changetype: modify
add: aci
aci: (targetattr="userPassword")(version 3.0; acl "Allow users to update
 their own password"; allow (write) userdn="ldap:///self";)
```

What's New

The following features are new with this release of Delegated Admin:

- Added the ability for developers to customize attribute form fields and form behaviors. Previously, `Delegated User Admin` did not support attributes whose types were neither strings nor numbers, like JSON attribute types. With the ability to override the form fields for any attribute type, developers can now extend `Delegated User Admin` to handle these attribute types and more. Requires PingDirectory Server 7.3.0.0 or later.
- Added more capabilities for attributes that are constructed from other attributes. The *constructed attribute* feature can now be used for attributes that the LDAP schema requires. Further, administrators can configure constructed attributes to update automatically when their referenced attributes change. For example, if `displayName` is constructed from `givenName` and `sn`, the `displayName` attribute updates whenever the delegated administrator updates either `givenName` or `sn`. Requires PingDirectory Server 7.3.0.1 or later.

Known Issues and Workarounds

The following issues are known in this release of Delegated Admin:

- Changes to custom form field HTML files are not displayed.

  This issue relates to the browser and server. Because PingDirectory Server does not send the appropriate cache headers for static files, some browsers might cache them aggressively.

  Workaround: Refer to the "Next steps" section in *Customizing UI form fields* on page 1266.
- Constructed attributes do not update automatically when editing a resource.

  This issue relates to the user interface (UI). When a constructed attribute is computed and saved in PingDirectory Server, the UI does not immediately display the new value.

  Workaround: To verify the changes visually, try one or both of the following suggestions:

  - Refresh the browser.
  - Visit another page, and then navigate back to the edited resource.

Resolved Issues

The following table identifies issues that have been resolved with this release of Delegated Admin.

Ticket ID	Description
DS-39347	Fixed an issue in which Delegated Admin did not work properly if the name of the REST resource type was not the same as the resource endpoint.
DS-39693	Fixed an issue in which Delegated Admin search results were truncated and invalid upon encountering a Directory entry that contained a Boolean or Integer syntax attribute whose values did not conform to the appropriate syntax. With this fix, the offending values are omitted from the results, and a warning message is logged to the server errors log.

PingDataSync Server Release Notes

PingDataSync 8.2.0.8 release notes

The release notes for the 8.2.0.8 release of the PingDataSync server.

Critical fixes

This release of the PingDataSync server addresses critical issues from earlier versions. Update all affected servers appropriately.

No critical issues have been identified.

Resolved issues

The following issues have been resolved with this release of the PingDataSync server.

Ticket ID	Description
DS-45164	Updated Jetty Server to version 9.4.44.
DS-45813	Updated PingDirectory products to use Kafka v2.8.1.

PingDataSync Server 8.2.0.7 release notes

The release notes for the 8.2.0.7 release of PingDataSync Server.

Critical fixes

This release of PingDataSync Server addresses critical issues from earlier versions. Update all affected servers appropriately.

- Fixed an issue where secret keys under `cn=Topology,cn=config` could be lost when removing a server from the topology. When a server is removed via the `dsreplication disable` or **`remove-defunct-server`** tools, its secret keys will now be distributed among the remaining members of the topology. The keys from the rest of the topology will also be copied to the server being removed.

  The cipher secret keys in the topology that are affected by this change are used by reversible password storage schemes (except for AES256, which uses the encryption settings database). If you are using a reversible password storage scheme other than AES256, prior to this fix, you could lose access to keys that had been used for reversible password encryption when removing servers from the topology.

  > ⓘ **Note:**
  >
  > Since this change only applies to the most recent version of **`remove-defunct-server`** and `dsreplication disable`, if you are removing a server from a multi-version topology, you should run that tool from the most recent version. In the past **`dsreplication`** and **`remove-defunct-server`** could only be run from an older version, but now in the case of removing a server from the topology, they should be run from the most recent version in the topology. If you run the tool from an older server, it will not include this fix, and you might lose access to secret keys from servers that are removed from the topology.

  - Fixed in: 8.2.0.7
  - Introduced in: 7.0.0.0
  - Support identifiers: DS-44591

Resolved Issues

The following issues have been resolved with this release of the Data Sync Server.

Ticket ID	Description
DS-44591	Fixed an issue where secret keys under `cn=Topology,cn=config` could be lost when removing a server from the topology. When a server is removed via the `dsreplication disable` or **`remove-defunct-server`** tools, its secret keys will now be distributed among the remaining members of the topology. The keys from the rest of the topology will also be copied to the server being removed. The cipher secret keys in the topology that are affected by this change are used by reversible password storage schemes (except for AES256, which uses the encryption settings database). If you are using a reversible password storage scheme other than AES256, prior to this fix, you could lose access to keys that had been used for reversible password encryption when removing servers from the topology. ⓘ **Note:** Since this change only applies to the most recent version of **`remove-defunct-server`** and `dsreplication disable`, if you are removing a server from a multi-version topology, you should run that tool from the most recent version. In the past **`dsreplication`** and **`remove-defunct-server`** could only be run from an older version, but now in the case of removing a server from the topology, they should be run from the most recent version in the topology. If you run the tool from an older server, it will not include this fix, and you might lose access to secret keys from servers that are removed from the topology.
DS-45124	Removed `-XX:RefDiscoveryPolicy=1` from the default start-server Java arguments. In rare cases, this argument was related to segmentation faults in the JVM, especially when used with the G1 garbage collector.
DS-45190	Added support for the use of JDKs obtained through BellSoft.

PingDataSync Server 8.2.0.6 release notes

Critical fixes

This release of PingDataSync Server addresses critical issues from earlier versions. Update all affected servers appropriately.

No critical issues have been identified.

Resolved issues

The following issues have been resolved with this release of the DataSync Server.

Ticket ID	Description
DS-43632	Fixed an issue where the `format` field is omitted from the list of operational attribute schemas in the Directory REST API.

PingDataSync Server 8.2.0.5 release notes

Critical Fixes

This release of the PingDataSync Server addresses critical issues from earlier versions. Update all affected servers appropriately.

No critical issues have been identified.

Resolved issues

The following issues have been resolved with this release of the PingDataSync Server.

Ticket ID	Description
DS-44025	Fixed an issue where the server was incorrectly displaying an `Unknown vendor` warning when using JDKs obtained on Red Hat and Ubuntu systems.

PingDataSync 8.2.0.2 release notes

Critical Fixes

This release of PingDataSync addresses critical issues from earlier versions. Update all affected servers appropriately.

No critical issues have been identified

Resolved Issues

The following issues have been resolved with this release of PingDataSync:

Ticket ID	Description
DS-43224	Made a generic OpenID Connect ID token validator available. This change allows single sign-on to the Administrative Console with OIDC providers other than just PingOne.

Ticket ID	Description
DS-43941	You can now specify that the Administrative Console use a custom truststore when evaluating OIDC provider certificates by using the oidc-trust-store-file and oidc-trust-store-type settings. Also, you can set the console to skip hostname and/or certification verification through the oidc-strict-hostname-verification and oidc-trust-all configuration settings.

PingDataSync 8.2.0.1 release notes

Critical Fixes

This release of PingDataSync addresses critical issues from earlier versions. Update all affected servers appropriately.

No critical issues have been identified

PingDataSync 8.2 release notes

Enhancements

The new feature for this release is:

- The Administrative Console now supports using OpenID Connect for admin SSO, allowing you to set up the PingOne administration console to have one-click SSO access without typing a password.

Upgrade considerations

Upgrade considerations are no longer part of the release notes. That information is now in *Upgrade overview and considerations* on page 1292.

Known issues and limitations

The following are known issues in the current version of Data Sync Server:

When signing on to the Administrative Console, you must specify an LDAPS port in the **Server** field. For example, to use port 636, you would specify pingdirectory:636. If you do not specify a port, you get a

```
"Server
      unavailable"
```

message.

Resolved Issues

The following issues have been resolved with this release of the Data Sync Server:

Ticket ID	Description
DS-5040	Improved the **`dsframework`** tool to support multivalued server properties.

Ticket ID	Description
DS-5143, DS-11035	Updated support for logging access and error log messages to a syslog server. While the server previously supported logging these messages to a syslog server (through the "syslog-based access log publisher" and "syslog-based error log publisher" logger implementations), these loggers used an older version of the syslog protocol (described in RFC 3164) and only offered support for communicating over UDP. These loggers are still available for legacy backward compatibility, but we now also offer new "syslog text access log publisher" and "syslog text error log publisher" implementations that use a newer version of the syslog protocol (syslog version 1, described in RFC 5424) and support communicating over UDP or the more reliable TCP. When using TCP, it is also possible to encrypt communication with TLS, and it is possible to configure multiple servers for better redundancy. These loggers use the same space-delimited text format as the former loggers. We also offer new "syslog JSON access log publisher" and "syslog JSON error log publisher" implementations that offer the same set of capabilities, but that format the message text as JSON objects, which can be more easily parsed by third-party software.
DS-10320, DS-12550, DS-12551, DS-12552, DS-42116, DS-42162, DS-42179, DS-42222, DS-42223, DS-42224, DS-42225, DS-42416, DS-42437	Added a config/sample-dsconfig-batch-files directory with set of well commented dsconfig batch files that may be useful in enabling or configuring a variety of features in the server.
DS-10775	Updated the dictionary password validator to support additional options: * It can now ignore non-alphabetic characters that appear at the beginning or end of the password before checking the dictionary. * It can strip characters of diacritical marks, including accents, cedillas, circumflexes, diaereses, tildes, and umlauts, before checking the dictionary. If this option is used, then any character with such a mark will be replaced with a base version of the character without that mark (for example, a lowercase letter n with a tilde over it would be replaced with just a lowercase letter n). * You can define maps with information about character substitutions to use for checking alternative versions of the provided password. For example, if you indicate that "0" might map to "o", "1" or "!" might map to "i", "7" might map to "t", and "3" might map to "e", then the validator can reject a proposed password of "pr0h1b!73d" if the dictionary contains the word "prohibited". * It can reject a proposed password if a value from the provided dictionary makes up more than a specified percentage of that password.

Ticket ID	Description
DS-11524, DS-41860, DS-42112	Added support for new administrative alert types: * We have added a new admin alert account status notification handler, which can generate administrative alerts whenever an applicable account status notification is generated within the server. For example, this account status notification handler can be added to the root password policy to generate an alert whenever a root user's password is updated or their account is locked as a result of too many failed authentication attempts. A separate alert type has been defined for each account status notification type. * We have added a new "privilege-assigned" administrative alert that can be raised whenever a new entry is added or an existing entry is updated to include one or more privileges. * We have added a new "insecure-request-rejected" administrative alert that can be raised whenever the server rejects a request as a result of the reject-insecure-requests global configuration property.
DS-13853	Added support for the OAUTHBEARER SASL mechanism (as described in RFC 7628) to allow LDAP clients to authenticate with OAuth 2.0 bearer tokens.
DS-15848, DS-42360	Added support for invoking a specified set of password validators during bind operations. If the password used to authenticate fails to satisfy one or more of the configured validators, the bind attempt can be rejected, the user can be forced to change their password, or the server can generate an account status notification to take some alternative action (for example, notifying the end user or server administrators).
DS-15864	Replaced the ldappasswordmodify tool with a new version that offers more functionality, including support for additional controls, support for multiple password change methods (the password modify extended operation, a regular LDAP modify operation, or an Active Directory-specific modify operation), and the ability to generate the new password on the client.
DS-16228	Added support for JSON-formatted Sync loggers and Sync Failed Ops loggers, which complement the existing file-based Sync logger and file- based Sync Failed Ops logger, respectively. The JSON-formatted Sync log messages provide information about synchronization events the server has processed, and can be written to files on the local filesystem, written to the JVM's standard output or standard error stream, or sent to a syslog server.
DS-17903	Updated setup to provide a --populateToolPropertiesFile argument that will allow it to populate the config/tools.properties file with default values for command-line tool arguments. If requested, properties will be provided for the server address, port, and communication security, and may also include a default bind DN and optionally a bind password. When running setup interactively, it will now prompt to determine which properties (if any) should be populated in the properties file.
DS-36088	Updated the crypto manager to make it possible to augment the set of enabled TLS cipher suites with specific suites to add to or remove from the default set of enabled suites. To enable one or more suites in addition to those in the default set, prefix the names of those suites with the "+" symbol. To disable one or more suites in the default set of enabled suites, prefix the names of those suites with the "-" symbol. This was already possible when configuring cipher suites for the LDAP and HTTP connection handlers, but it was not an option for the crypto manager.

Ticket ID	Description
DS-37890	Added a new property to sync classes called replace-all-attr-values- limit. The property is only used when replace-all-attr-values is true and the sync destination is an LDAP server. If a modification would replace more values of the same attribute than the value of replace-all- attr-values-limit, then the modification will use ADD or DELETE operations rather than a REPLACE operation.
DS-38110	Updated the System Information monitor with an "isDocker" attribute to identify if the server is running in a Docker container.
DS-38118, DS-42495	Made several updates related to the server's handling of data written to standard output and standard error: * The server can now be configured to rotate the logs/server.out file once it reaches a given size, and it will retain a configurable number of those log files. By default, the server will rotate the file once it reaches 100 megabytes and will keep up to ten files. * To better facilitate capturing log data in containerized environments, the server now supports writing JSON-formatted access and error log messages to the JVM's original standard output and error streams (which will be separate from the server.out file when the server is started with the --nodetach argument). * It is now possible to prevent the server from logging messages during startup in non-JSON format. It is also possible to prevent messages about administrative alerts from being written to standard error, or to write those messages in JSON format. These options are especially useful when using JSON-based logging to the console in no-detach mode, as they can help ensure that everything written to standard output and standard error will be formatted as JSON objects.
DS-38868	Updated setup to create a second encryption settings definition if data encryption is enabled. It will continue to create a definition for 128-bit AES encryption for use as the preferred definition to preserve backward compatibility with existing servers in the topology, but it will now also generate a definition for 256-bit AES encryption. The 256-bit AES definition may become the preferred definition in a future release, but you can use it now by first ensuring that any existing instances are updated to contain the new definition (with the "encryption-settings export" and "encryption-settings import" commands) and then making it the preferred definition (with "encryption-settings set-preferred") in all instances.
DS-39646	Updated the manage-topology add-server command to configure failover for Data Sync servers when they are added to a topology.
DS-39789	Updated the JVM memory usage monitor provider to fix an issue that could prevent the monitor from reporting the total amount of memory held by all memory consumers. Also, fixed an issue that could cause the memory-consumer attribute to use an incomplete message for consumers without a defined maximum size and added an additional memory-consumer-json attribute whose values are JSON objects with data that can be more easily extracted by automated processes.
DS-40650	Updated the collect-support-data tool to make it possible to specify how much data should be captured from the beginning and end of each log file to include in the support data archive. You can also specify the capture size when invoking the tool through an administrative task, recurring task, or extended operation.
DS-40828	Fixed an issue where some state associated with a JMX connection was not freed after the connection was closed. This led to a slow memory leak in servers that were monitored by an application that created a new JMX connection each polling interval.

Ticket ID	Description
DS-40967	Eliminated a misleading error message that could be logged at startup if the server was configured with one or more ACIs that only apply when using specific SASL mechanisms.
DS-41350	Fixed an issue where disabling certain backends (such as 'alarms') caused an internal monitor to log unnecessary error messages every few seconds, about not being able to gather data from that backend. Note that deliberately disabling the 'alarms' backend is not recommended in normal operation, but may occur during backup/restore operations.
DS-41807	Fixed an issue where LDAP DELETE operations were sometimes not synchronized from an Oracle Unified Directory sync source, due to variations in the format of the 'targetUniqueID' attribute obtained from the Oracle changelog.
DS-41964	Fixed an issue with the manage-profile tool where files in a server profile's dsconfig/ directory without a ".dsconfig" extension could cause failures in manage-profile replace-profile when validating updated dsconfig files.
DS-41989	Fixed an issue that could result in duplicate column headers being produced by the Periodic Stats Logger, even when the header-prefix-per-column attribute was set to true.
DS-42004	Added a new configuration property, changes-queue-size, that controls the size of the changes queue on a Sync Pipe. This was previously only configurable via a system property, and it would change the queue size of all Sync Pipes. DataSync will continue to accept the system property, but it is encouraged to migrate the setting into the configuration.
DS-42045	Updated the Stats Collector Plugin with a new generate-collector-files configuration property. When using the plugin exclusively for providing metrics to one or more StatsD Monitoring Endpoints, set this property to false to prevent unnecessary I/O.
DS-42049	Fixed an issue with Sync server where retry ops could get the server stuck and unable to process changes at the source.
DS-42059, DS-42060	Updated setup to add options for improving communication security: * Non-interactive setup now offers a --rejectInsecureRequests argument that will configure the server to reject any request received over a connection that is not encrypted with SSL or StartTLS. * Non-interactive setup now offers a --rejectUnauthenticatedRequests argument that will configure the server to reject any request received over a connection that is not authenticated (or that is authenticated as the anonymous user). * Interactive setup now allows you to configure the server with the LDAP connection handler disabled (which was already an option when using non-interactive setup), or enabled but only for communication encrypted with StartTLS. The --rejectInsecureRequests and --rejectUnauthenticatedRequests arguments can also be used with manage-profile by including them in the setup-arguments.txt file of the server profile.

Ticket ID	Description
DS-42061	Updated the interactive command-line tool framework to prefer establishing secure LDAP connections over insecure connections. Previously, when prompting for the information needed to establish a connection, the default option was to create an unencrypted LDAP connection. Now, tools will default to creating an SSL-encrypted connection if the server supports it, or to creating a StartTLS-encrypted connection if that is available but SSL is not. Tools will also default to using streamlined settings when establishing secure connections. Previously, they would always prompt about how to determine whether the server's certificate chain should be trusted. When using the streamlined settings, the tools will only prompt about certificates that cannot automatically be considered trusted using information in the JVM's default trust store, the server's default trust store (config/truststore), or the server's topology registry.
DS-42062	Updated the root password policy so that LDAP bind responses for root users and topology administrators will be delayed by one second after five consecutive failed authentication attempts.
DS-42063	Updated the "delay bind response" failure lockout action to provide an option to delay the response to bind requests initiated by non-LDAP clients (for example, when using HTTP basic authentication). This option is disabled by default because delaying the bind response for non-LDAP clients may require temporarily blocking the thread used to process the request, which could increase the risk of a denial-of-service attack. To help mitigate this risk, if you enable delayed bind responses for non-LDAP clients, we recommend that you also increase the number of request handler threads for all enabled HTTP connection handlers.
DS-42115	Updated the server's command-line tool framework to make it easier and more convenient to communicate with the server over a secure connection when no trust-related arguments are provided. Most non-interactive tools will now check the server's default trust store, the topology registry, and the JVM's default trust store to see if the presented certificate chain can be automatically trusted without the need to prompt the user. If the presented chain cannot be automatically trusted, the user may be interactively prompted to determine whether it should be trusted.
DS-42199	Optimized some searches commonly used by the status tool. This should improve the performance of the tool in more complex or large-scale environments.
DS-42265	Upgrade to jetty 9.4
DS-42276	Fixed an issue where using the encryption-settings tool to import definitions with the set-preferred flag could result in none of the imported definitions being set as the preferred definition.
DS-42279	Updated the server to require a minimum key size of 2048 bits when negotiating a TLS cipher suite that uses ephemeral Diffie-Hellman key exchange.
DS-42298	Replaced the ldifsearch, ldifmodify, and ldif-diff command-line tools with more full-featured and robust implementations.
DS-42331	Replaced the ldapcompare tool with a new version that offers more functionality, including support for multiple compare assertions, following referrals, additional controls, and multiple output formats (including tab-delimited text, CSV, and JSON).

Ticket ID	Description
DS-42347	Updated the server to use /dev/urandom (on non-Windows systems where that path exists and is readable) instead of /dev/random as the primary source for secure random data. Attempts to read from /dev/random can block if the underlying system does not have sufficient entropy, which can have a severe adverse effect on performance. Reads from /dev/urandom will not block, and the data that it provides is no less secure than data from /dev/random in any way that matters for the server.
DS-42349, DS-43209, DS-43210, DS-43323, DS-43324	Added support for JSON-formatted audit loggers, which complement the existing file-based LDIF-formatted error logger. The JSON-formatted audit log messages provide a record of changes to data in the server and can be written to files on the local filesystem, written to the JVM's standard output or standard error stream, or sent to a syslog server. Added support for JSON-formatted HTTP operation loggers, which complement the existing file-based loggers using the W3C common log format and a proprietary space-delimited text format. The JSON-formatted HTTP operation log messages provide a record of interaction with HTTP clients and can be written to files on the local filesystem, written to the JVM's standard output or standard error stream, or sent to a syslog server. Fixed an issue that caused JSON-formatted loggers to use a timestamp format that was not strictly compliant with the ISO 8601 format described in RFC 3339. Timestamps incorrectly omitted the colon between the hour and minute components of the time zone offset.
DS-42504	Updated manage-profile replace-profile to set encryption settings definitions defined in the newer server profile as preferred in the encryption settings db.
DS-42547	Fixed an issue where manage-profile generate-profile would print "null" as the generated profile directory when writing to an existing directory.
DS-42609	Fixed an issue in which the Directory REST API could fail to decode certain credentials when using basic authentication.
DS-42632	Added support for creating or importing a key pair configuration object using an elliptic curve (EC) key algorithm. You can use this to designate the encryption key pair for a JWT access token validator that handles EC-encrypted access tokens.
DS-42634	The JWT Access Token Validator can now validate JWT access tokens signed using the elliptic curve digital signature algorithms ES256, ES384, and ES512.
DS-42635	The JWT Access Token Validator can now validate JWT access tokens encrypted using elliptic curve cryptographic algorithms. The following key encryption algorithms are now supported in addition to RSA-OAEP: ECDH-ES, ECDH-ES+A128KW, ECDH-ES+A192KW, and ECDH-ES +A256KW. To support best practices for JWT security, you must now also configure the JWT Access Token Validator with explicit allow lists for key encryption and content encryption algorithms. For backward compatibility, the key encryption allow list defaults to RSA-OAEP, while the content encryption allow list defaults to A128CBC-HS256, A192CBC-HS384, and A256CBC-HS512. We recommend setting both allow lists to the strict minimum set of algorithms needed by the Access Token Validator.
DS-42651	Updated the manage-profile replace-profile subcommand to better support updating the server's keystore and truststore files. When using the --generateSelfSignedCertificate argument in a server profile's setup-arguments.txt file, the server will maintain the original keystore and truststore files during replace-profile. Otherwise, replace-profile will use the keystore and truststore specified in the profile's setup-arguments.txt file.

Ticket ID	Description
DS-42665	Fixed an issue that caused the resync tool to not take the attribute-comparison-method of the sync class into account. This caused resync to ignore when byte-for-byte comparison was configured.
DS-42667	Updated the server to set a unique cluster name when started for the first time.
DS-42669, DS-42748	Updated the online dsconfig step of the manage-profile replace-profile subcommand to support getting LDAP connection arguments from a tools.properties file on the server being updated. Fixed an issue where boolean LDAP connection arguments like --useSSL and --trustAll would cause manage-profile replace-profile to fail when applying dsconfig online.
DS-42673	Updated the manage-profile setup subcommand to fail if the start-server command has a non-zero exit code.
DS-42681, DS-42684	Performance statistics generated by the Sideband API can now be published by the Periodic Stats Logger. To enable this, use the "included-http-servlet-stat" property of the Periodic Stats Logger.
DS-42687	Upgrade to Jetty 9.4.30
DS-42740	Fixed an issue where the **`dsconfig list`** subcommand would not display requested properties.
DS-42749	To support best practices for JWT security, you must now configure the JWT Access Token Validator with an explicit list of the JWT signing algorithms that it accepts. For backward compatibility, this list defaults to the RSA signing algorithms RS256, RS384, and RS512, but we recommend setting this list to the strict minimum set of signing algorithms needed by the Access Token Validator.
DS-42751	Added new override-status-code and additional-response-contents attributes to the Availability State HTTP Servlet Extension. These new attributes can be used to customize the response code and JSON response body of the servlet.
DS-42850	Fixed a typo in the password-expiring template that caused "password_expiration_time_of_day" to be printed instead of the password expiration time.
DS-42861	Updated the manage-profile tool logs to include the duration of each step the tool takes. The new --verbose argument can also be used to display timing information in the tool's console output.
DS-42872	Added a JSON-formatted stats logger to the server's default configuration. The stats logger is disabled by default.
DS-42886	Updated non-interactive setup (including manage-profile setup) to allow the password for the initial root user to be provided in pre-encoded form using the PBKDF2, SSHA256, SSHA384, or SSHA512 password storage scheme. This eliminates the need to have access to the clear-text password when setting up the server.
DS-42926	Fixed an issue where Ping Directory products configured to run as Microsoft Windows services were sometimes unable to automatically restart following an unplanned reboot, due to errors reading a corrupted server status file.

Ticket ID	Description
DS-42939	The Administrative Console configuration settings have been updated to account for the new SSO functionality.
DS-42952	For Windows only, there can be a hang on start when global configuration property `startup-error-logger-output-location` is set to values that contain `standard-error`. For Windows only, `standard-error` values are silently mapped to equivalent `standard-output` values.
DS-42963	Updated the manage-profile generate-profile subcommand to ignore files larger than 100 megabytes when generating a server profile. Fixed an issue where many large files in the server root could cause the tool to run out of memory.
DS-43027	Added a new --adminPasswordFile argument to the manage-topology add-server command, to allow specifying the administrator password with a file rather than with the command line.
DS-43073, DS-43198	Added support for ID Token Validators, which validate the integrity and content of ID tokens issued by OpenID Connect providers. Use these validators with the OAuth Bearer SASL Mechanism Handler to enable single sign-on (SSO) for the Administrative Console using an OpenID Connect provider such as PingOne. Currently, only PingOne is supported for SSO.
DS-43074	Added three built-in identity mappers that you can use to look up administrative accounts stored in the server configuration: Root DN Users, Topology Admin Users, and All Admin Users.
DS-43288	Updated setup and the replace-certificate tool to improve the way we generate self-signed certificates and certificate signing requests to make them more palatable to clients. To reduce the frequency with which administrators had to replace self-signed certificates, we previously used a very long lifetime for self-signed certificates generated by setup or the replace-certificate tool. However, some clients (especially web browsers and other HTTP clients) have started more strenuously objecting to certificates to long lifetimes, so we now generate self-signed certificates with a one-year validity period. The inter-server certificate (which is used internally within the server and does not get exposed to normal clients) is still created with a twenty-year lifetime. Also, the replace-certificate tool's interactive mode has been updated to improve the process that it uses to obtain information to include in the subject DN and subject alternative name extension for self-signed certificates and certificate signing requests. The following changes have been made in accordance with CA/Browser Forum guidelines: * When selecting the subject DN for the certificate, we listed a number of common attributes that may be used, including CN, OU, O, L, ST, and C. We previously indicated that CN attribute was recommended. We now also indicate that the O and C attributes are recommended as well. * When obtaining the list of DNS names to include in the subject alternative name extension, we previously suggested all names that we could find associated with interfaces on the local system. In many cases, we now omit non-qualified names and names that are associated with loopback interfaces. We will also warn about any attempts to add unqualified or invalid names to the list. * When obtaining the list of IP addresses to include in the subject alternative name extension, we previously suggested all addresses associated with all network interfaces on the system. We no longer suggest any IP addresses associated with loopback interfaces, and we no longer suggest any IP addresses associated in IANA-reserved ranges (for example, addresses reserved for private-use networks). The tool will now warn about attempts to add these addresses for inclusion in the subject alternative name extension.

Ticket ID	Description
DS-43305	Increased the maximum number of RDN components that a DN may have from 50 to 100.
DS-43376	Updated log publisher logic to reduce the amount of CPU that the server consumes when it is idle.
DS-43480	Updated the system information monitor provider to restrict the set of environment variables that may be included. Previously, the monitor entry included information about all defined environment variables, as that information can be useful for diagnostic purposes. However, some deployments may include credentials, secret keys, or other sensitive information in environment variables, and that should not be exposed in the monitor. The server will now only include values from a predefined set of environment variables that are expected to be the most useful for troubleshooting problems, and that are not expected to contain sensitive information.
DS-43517	Updated the jose4j library used for JWT signing and encryption to version 0.7.2.
DS-43651	The Security Guide is now available online at pingidentity.com. The guide has been removed from the server packaging.

Critical fixes

Critical Fixes

This release of the Data Sync Server addresses critical issues from earlier versions. Update all affected servers appropriately.

- Addressed an issue that could lead to slow, off-heap memory growth. This only occurred on servers whose cn=Version,cn=monitor entry was retrieved frequently.
  - Fixed in: 8.1.0.0
  - Introduced in: 5.2.0.0
  - Support identifiers: DS-41301
- Fixed a memory leak when performing SCIM queries on the Directory Server.
  - Fixed in: 8.1.0.0
  - Introduced in: 7.2.0.0
  - Support identifiers: DS-41206 SF#00681395
- The following enhancements were made to the topology manager to make it easier to diagnose connection errors:

  - Added monitoring information for all the failed outbound connections (including the time since it's been failing and the last error message seen when the failure occurred) from a server to one of its configured peers and the number of failed outbound connections.

  - Added alarms/alerts for when a server fails to connect to a peer server within a configured grace period.

  - Fixed in: 7.3.0.0
  - Introduced in: 7.0.0.0
  - Support identifiers: DS-38334 SF#00655578
- The topology manager will now raise a mirrored-subtree-manager-connection-asymmetry alarm when a server is able to establish outbound connections to its peer servers, but those peer servers are unable

to establish connections back to the server within the configured grace period. The alarm is cleared as soon as there is connection symmetry.

  - Fixed in: 7.3.0.0
  - Introduced in: 7.0.0.0
  - Support identifiers: DS-38344 SF#00655578
- The dsreplication tool has been fixed to work when the node being used to enable replication is currently out-of-sync with the topology master.
  - Fixed in: 7.3.0.0
  - Introduced in: 7.0.0.0
  - Support identifiers: DS-38335 SF#00655578
- Fixed two issues in which the server could have exposed some clear-text passwords in files on the server file system.

  * When creating an encrypted backup of the alarms, alerts, configuration, encryption settings, schema, tasks, or trust store backends, the password used to generate the encryption key (which may have been obtained from an encryption settings definition) could have been inadvertently written into the backup descriptor. This problem does not affect local DB backends (like userRoot), the LDAP changelog backend, or the replication database.

  * When running certain command-line tools with an argument instructing the tool to read a password from a file, the password contained in that file could have been written into the server's tool invocation log instead of the path to that file. Affected tools include backup, create-initial-config, create-initial-proxy-config, dsreplication, enter-lockdown-mode, export-ldif, import-ldif, ldappasswordmodify, leave-lockdown-mode, manage-tasks, manage-topology, migrate-ldap-schema, parallel-update, prepare-endpoint-server, prepare-external-server, realtime-sync, rebuild-index, re-encode-entries, reload-http-connection-handler-certificates, reload-index, remove-defunct-server, restore, rotate-log, and stop-server. Other tools are not affected. Also note that this only includes passwords contained in files that were provided as command-line arguments; passwords included in the tools.properties file, or in a file referenced from tools.properties, would not have been exposed.

  In each of these cases, the files would have been written with permissions that make their contents only accessible to the system account used to run the server. Further, while administrative passwords may have been exposed in the tool invocation log, neither the passwords for regular users, nor any other data from their entries, should have been affected. We have introduced new automated tests to help ensure that such incidents do not occur in the future.

  We recommend changing any administrative passwords you fear may have been compromised as a result of this issue. If you are concerned that the passphrase for an encryption settings definition may have been exposed, then we recommend creating a new encryption settings definition that is preferred for all subsequent encryption operations, exporting your data to LDIF, and re-importing so that it will be encrypted with the new key. You also may wish to re-encrypt or destroy any existing backups, LDIF exports, or other data encrypted with a compromised key, and you may wish to sanitize or destroy any existing tool invocation log files that may contain clear-text passwords.

  - Fixed in: 7.3.0.0
  - Introduced in: 7.0.0.0
  - Support identifiers: DS-38897 DS-38908

- The following enhancements were made to the topology manager to make it easier to diagnose the connection errors:

  - Added monitoring information for all the failed outbound connections (including the time since it's been failing and the last error message seen when the failure occurred) from a server to one of its configured peers and the number of failed outbound connections.

  - Added alarms/alerts for when a server fails to connect to a peer server within a configured grace period.

  - Fixed in: 7.2.1.0
  - Introduced in: 7.0.0.0
  - Support identifiers: DS-38334 SF#00655578
- The topology manager will now raise a mirrored-subtree-manager-connection-asymmetry alarm when a server is able to establish outbound connections to its peer servers, but those peer servers are unable to establish connections back to the server within the configured grace period. The alarm is cleared when connection symmetry is achieved.

  - Fixed in: 7.2.1.0
  - Introduced in: 7.0.0.0
  - Support identifiers: DS-38344 SF#00655578
- The dsreplication tool has been fixed to work when the node being used to enable replication is currently out-of-sync with the topology master.

  - Fixed in: 7.2.1.0
  - Introduced in: 7.0.0.0
  - Support identifiers: DS-38335 SF#00655578
- Fixed two issues in which the server could have exposed some clear-text passwords in files on the server file system.

  * When creating an encrypted backup of the alarms, alerts, configuration, encryption settings, schema, tasks, or trust store backends, the password used to generate the encryption key (which may have been obtained from an encryption settings definition) could have been inadvertently written into the backup descriptor. This problem does not affect local DB backends (like userRoot), the LDAP changelog backend, or the replication database.

  * When running certain command-line tools with an argument instructing the tool to read a password from a file, the password contained in that file could have been written into the server's tool invocation log instead of the path to that file. Affected tools include backup, create-initial-config, create-initial-proxy-config, dsreplication, enter-lockdown-mode, export-ldif, import-ldif, ldappasswordmodify, leave-lockdown-mode, manage-tasks, manage-topology, migrate-ldap-schema, parallel-update, prepare-endpoint-server, prepare-external-server, realtime-sync, rebuild-index, re-encode-entries, reload-http-connection-handler-certificates, reload-index, remove-defunct-server, restore, rotate-log, and stop-server. Other tools are not affected. Also note that this only includes passwords contained in files that were provided as command-line arguments; passwords included in the tools.properties file, or in a file referenced from tools.properties, would not have been exposed.

  In each of these cases, the files would have been written with permissions that make their contents only accessible to the system account used to run the server. Further, while administrative passwords may have been exposed in the tool invocation log, neither the passwords for regular users, nor any other data from their entries, should have been affected. We have introduced new automated tests to help ensure that such incidents do not occur in the future.

  We recommend changing any administrative passwords you fear may have been compromised as a result of this issue. If you are concerned that the passphrase for an encryption settings definition may have been exposed, then we recommend creating a new encryption settings definition that is preferred for all subsequent encryption operations, exporting your data to LDIF, and re-importing so that it will be encrypted with the new key. You also may wish to re-encrypt or destroy any existing backups, LDIF

exports, or other data encrypted with a compromised key, and you may wish to sanitize or destroy any existing tool invocation log files that may contain clear-text passwords.

  - Fixed in: 7.0.1.3
  - Introduced in: 7.0.0.0
  - Support identifiers: DS-38897 DS-38908
- Fixed an issue with the sync connect and response timeouts being set with incorrect units of time.
  - Fixed in: 6.2.0.0
  - Introduced in: 6.0.0.0
  - Support identifiers: DS-18026 SF#00616763
- Fixed an issue with the sync connect and response timeouts being set with incorrect units of time.
  - Fixed in: 6.2.0.0
  - Introduced in: 6.0.0.0
  - Support identifiers: DS-18026 SF#00616763
- The server can now detect an "out of file handles" situation on the operating system, and shut down to prevent running in an unreliable state.
  - Fixed in: 5.1.0.0
  - Introduced in: 2.1.0.0
  - Support identifiers: DS-12579 SF#2655
- Disabled support for SSLv3 by default in the LDAP, HTTP, and JMX connection handlers, and for replication communication. The recently-discovered POODLE vulnerability could potentially allow a network attacker to determine the plaintext behind an SSLv3-encrypted session, which would effectively negate the primary benefit of the encryption.

  SSLv3 was initially defined in 1996, but was supplanted by the release of the TLSv1 definition in 1999 (and subsequently by TLSv1.1 in 2006 and TLSv1.2 in 2008). These newer TLS protocols are not susceptible to the POODLE vulnerability, and the server has supported them (and preferred them over SSLv3) for many years. The act of disabling SSLv3 by default should not have any adverse effect on clients that support any of the newer TLS protocols. However, if there are any legacy client applications that attempt to communicate securely but do not support the newer TLS protocols, they should be updated to support the newer protocols. In the event that there are known clients that do not support any security protocol newer than SSLv3 and that cannot be immediately updated to support a newer protocol, SSLv3 support can be re-enabled using the newly-introduced allowed-insecure-tls-protocol global configuration property. However, since communication using SSLv3 can no longer be considered secure, it is strongly recommended that every effort be made to update all known clients still using SSLv3.

  It is possible to use the server access log to identify LDAP clients that use SSLv3 to communicate with the server. Whenever an LDAP client establishes a secure connection to the server, or whenever a client uses the StartTLS extended operation to secure an existing plaintext connection, the server will generate a SECURITY-NEGOTIATION access log message. The "protocol" element of a SECURITY-NEGOTIATION access log message specifies the name of the security protocol that has been negotiated between the client and the server, and any SECURITY-NEGOTIATION messages with a protocol of "SSLv3" suggest that the associated client is vulnerable to the POODLE attack. In addition, if any connections are terminated for attempting to use the disallowed SSLv3 protocol, the access log message for that disconnect should include a message stating the reason for the termination.

  - Fixed in: 5.0.0.0
  - Introduced in: 2.1.0.0
  - Support identifiers: DS-11782
- Change the default behavior of the Synchronization Server to not lock entries across all Sync Pipes when processing changes.

  The Sync Server has a specialized mutex that ensures that changes to the same entry are processed serially. The primary reason for this mutex is to ensure that the server can safely process changes

in parallel to achieve high throughput. However, we also use this mutex to ensure that two Sync Pipes don't process the same entry at the same time for deployments that synchronize changes bi-directionally. A consequence of this locking is that if one Sync Pipe is failing (because the destination is unavailable) then it retains the lock on an entry, and when other Sync Pipes try to process changes to that entry they will block that change and all changes that follow it while they wait on the lock.

This change turns off using a shared mutex by default, but adds a new advanced configuration option on the Sync Pipe, shared-mutex-name, that specifies the name of a mutex that is shared by other Sync Pipes. This gives greater control over the locking so that two Sync Pipes that share end points can ensure that two changes to the same logical user are not processed concurrently, while not impacting other Sync Pipes.

See the shared-mutex-name property for more information.

This property is subject to change in a future release.

- Fixed in: 3.2.0.0
- Introduced in: 3.0.0.0
- Support identifiers: DS-4202 SF#1527

PingDataSync Server Release Notes archive

Release Notes for earlier versions of PingDataSync Server are included for reference.

PingDataSync Server 8.1.0.6 release notes
The release notes for the 8.1.0.6 release of PingDataSync Server.

Critical fixes

This release of the Data Sync Server addresses critical issues from earlier versions. Update all affected servers appropriately.

No critical issues have been identified.

Resolved issues

The following issues have been resolved with this release of the Data Sync Server.

Ticket ID	Description
DS-42049	Fixed an issue with the Data Sync server where retry operations could get the server stuck and unable to process changes at the source.

PingDataSync Server 8.1.0.5 release notes

Critical fixes

This release of the Data Sync Server addresses critical issues from earlier versions. Update all affected servers appropriately.

No critical issues have been identified.

Resolved issues

The following issues have been resolved with this release of the Data Sync Server.

Ticket ID	Description
DS-43288	Updated setup and the replace-certificate tool to improve the way we generate self-signed certificates and certificate signing requests to make them more palatable to clients. To reduce the frequency with which administrators had to replace self-signed certificates, we previously used a very long lifetime for self-signed certificates generated by setup or the replace-certificate tool. However, some clients (especially web browsers and other HTTP clients) have started more strenuously objecting to certificates to long lifetimes, so we now generate self-signed certificates with a one-year validity period. The inter-server certificate (which is used internally within the server and does not get exposed to normal clients) is still created with a twenty-year lifetime. Also, the replace-certificate tool's interactive mode has been updated to improve the process that it uses to obtain information to include in the subject DN and subject alternative name extension for self-signed certificates and certificate signing requests. The following changes have been made in accordance with CA/Browser Forum guidelines: * When selecting the subject DN for the certificate, we listed a number of common attributes that may be used, including CN, OU, O, L, ST, and C. We previously indicated that CN attribute was recommended. We now also indicate that the O and C attributes are recommended as well. * When obtaining the list of DNS names to include in the subject alternative name extension, we previously suggested all names that we could find associated with interfaces on the local system. In many cases, we now omit non-qualified names and names that are associated with loopback interfaces. We will also warn about any attempts to add unqualified or invalid names to the list. * When obtaining the list of IP addresses to include in the subject alternative name extension, we previously suggested all addresses associated with all network interfaces on the system. We no longer suggest any IP addresses associated with loopback interfaces, and we no longer suggest any IP addresses associated in IANA-reserved ranges (for example, addresses reserved for private-use networks). The tool will now warn about attempts to add these addresses for inclusion in the subject alternative name extension.

Ticket ID	Description
DS-44316	Reduced the JVM memory requirements for many command line tools. This avoids memory pressure when multiple tools, such as a scheduled collect-support-data task, are run concurrently to the server process. For most tools, the initial heap size has been reduced to 128 MB, and for certain tools the maximum heap size has capped at 512 MB. On systems with larger amounts of memory, these tools previously were allotted unnecessarily large heaps. The maximum heap size has not been reduced for any tool that especially benefits from having more memory.

PingDataSync 8.1.0.2 Release Notes

Critical Fixes

This release of the Data Sync Server addresses critical issues from earlier versions. Update all affected servers appropriately.

- Addressed an issue that could lead to slow, off-heap memory growth. This only occurred on servers whose cn=Version,cn=monitor entry was retrieved frequently.
  - Fixed in: 8.1.0.0
  - Introduced in: 5.2.0.0
  - Support identifiers: DS-41301
- Fixed a memory leak when performing SCIM queries on the Directory Server.
  - Fixed in: 8.1.0.0
  - Introduced in: 7.2.0.0
  - Support identifiers: DS-41206 SF#00681395
- The following enhancements were made to the topology manager to make it easier to diagnose connection errors:
  - Added monitoring information for all the failed outbound connections (including the time since it has been failing and the last error message seen when the failure occurred) from a server to one of its configured peers and the number of failed outbound connections.
  - Added alarms/alerts for when a server fails to connect to a peer server within a configured grace period.
  - Fixed in: 7.3.0.0
  - Introduced in: 7.0.0.0
  - Support identifiers: DS-38334 SF#00655578
- The topology manager will now raise a mirrored-subtree-manager-connection-asymmetry alarm when a server is able to establish outbound connections to its peer servers, but those peer servers are unable to establish connections back to the server within the configured grace period. The alarm is cleared as soon as there is connection symmetry.
  - Fixed in: 7.3.0.0
  - Introduced in: 7.0.0.0
  - Support identifiers: DS-38344 SF#00655578

- The dsreplication tool has been fixed to work when the node being used to enable replication is currently out-of-sync with the topology master.
  - Fixed in: 7.3.0.0
  - Introduced in: 7.0.0.0
  - Support identifiers: DS-38335 SF#00655578
- Fixed two issues in which the server could have exposed some clear-text passwords in files on the server file system.
  - Fixed in: 7.3.0.0
  - Introduced in: 7.0.0.0
  - Support identifiers: DS-38897 DS-38908
- The following enhancements were made to the topology manager to make it easier to diagnose the connection errors:
  - Fixed in: 7.2.1.0
  - Introduced in: 7.0.0.0
  - Support identifiers: DS-38334 SF#00655578
- The topology manager will now raise a mirrored-subtree-manager-connection-asymmetry alarm when a server is able to establish outbound connections to its peer servers, but those peer servers are unable to establish connections back to the server within the configured grace period. The alarm is cleared when connection symmetry is achieved.
  - Fixed in: 7.2.1.0
  - Introduced in: 7.0.0.0
  - Support identifiers: DS-38344 SF#00655578
- The dsreplication tool has been fixed to work when the node being used to enable replication is currently out-of-sync with the topology master.
  - Fixed in: 7.2.1.0
  - Introduced in: 7.0.0.0
  - Support identifiers: DS-38335 SF#00655578
- Fixed two issues in which the server could have exposed some clear-text passwords in files on the server file system.
  - Fixed in: 7.0.1.3
  - Introduced in: 7.0.0.0
  - Support identifiers: DS-38897 DS-38908
- Fixed an issue with the sync connect and response timeouts being set with incorrect units of time.
  - Fixed in: 6.2.0.0
  - Introduced in: 6.0.0.0
  - Support identifiers: DS-18026 SF#00616763
- Fixed an issue with the sync connect and response timeouts being set with incorrect units of time.
  - Fixed in: 6.2.0.0
  - Introduced in: 6.0.0.0
  - Support identifiers: DS-18026 SF#00616763
- The server can now detect an "out of file handles" situation on the operating system, and shut down to prevent running in an unreliable state.
  - Fixed in: 5.1.0.0
  - Introduced in: 2.1.0.0
  - Support identifiers: DS-12579 SF#2655
- Disabled support for SSLv3 by default in the LDAP, HTTP, and JMX connection handlers, and for replication communication. The recently-discovered POODLE vulnerability could potentially allow a

network attacker to determine the plaintext behind an SSLv3-encrypted session, which would effectively negate the primary benefit of the encryption.

  - Fixed in: 5.0.0.0
  - Introduced in: 2.1.0.0
  - Support identifiers: DS-11782
- Change the default behavior of the Synchronization Server to not lock entries across all Sync Pipes when processing changes.
  - Fixed in: 3.2.0.0
  - Introduced in: 3.0.0.0
  - Support identifiers: DS-4202 SF#1527

Resolved Issues

The following issues have been resolved with this release of the Data Sync Server:

Ticket ID	Description
DS-40828	Fixed an issue where some state associated with a JMX connection was not freed after the connection was closed. This led to a slow memory leak in servers that were monitored by an application that created a new JMX connection each polling interval.
DS-41964	Fixed an issue with the manage-profile tool where files in a server profile's dsconfig/ directory without a ".dsconfig" extension could cause failures in manage-profile replace-profile when validating updated dsconfig files.
DS-42687	Upgrade to Jetty 9.4.30

PingDataSync 8.1 Release Notes

Critical Fixes

This release of the Data Sync Server addresses critical issues from earlier versions. Update all affected servers appropriately.

- Addressed an issue that could lead to slow, off-heap memory growth. This only occurred on servers whose cn=Version,cn=monitor entry was retrieved frequently.
  - Fixed in: 8.1.0.0
  - Introduced in: 5.2.0.0
  - Support identifiers: DS-41301
- Fixed a memory leak when performing SCIM queries on the Directory Server.
  - Fixed in: 8.1.0.0
  - Introduced in: 7.2.0.0
  - Support identifiers: DS-41206 SF#00681395
- Fixed a memory leak when performing SCIM queries on PingDataMetrics Server.
  - Fixed in: 8.0.0.1

    Introduced in: 7.2.0.0

    Support identifiers: DS-41206 SF#00681395

- Addressed an issue that could lead to slow off-heap memory growth. This only occurred on servers whose cn=Version,cn=monitor entry was retrieved frequently.
  - Fixed in: 8.0.0.1

    Introduced in: 5.2.0.0

    Support identifiers: DS-41301
- The following enhancements were made to the topology manager to make it easier to diagnose connection errors:

  - Added monitoring information for all the failed outbound connections (including the time since it's been failing and the last error message seen when the failure occurred) from a server to one of its configured peers and the number of failed outbound connections.

  - Added alarms/alerts for when a server fails to connect to a peer server within a configured grace period.
  - Fixed in: 7.3.0.0
  - Introduced in: 7.0.0.0
  - Support identifiers: DS-38334 SF#00655578
- The topology manager will now raise a mirrored-subtree-manager-connection-asymmetry alarm when a server is able to establish outbound connections to its peer servers, but those peer servers are unable to establish connections back to the server within the configured grace period. The alarm is cleared as soon as there is connection symmetry.
  - Fixed in: 7.3.0.0
  - Introduced in: 7.0.0.0
  - Support identifiers: DS-38344 SF#00655578
- The dsreplication tool has been fixed to work when the node being used to enable replication is currently out-of-sync with the topology master.
  - Fixed in: 7.3.0.0
  - Introduced in: 7.0.0.0
  - Support identifiers: DS-38335 SF#00655578
- Fixed two issues in which the server could have exposed some clear-text passwords in files on the server file system.

  * When creating an encrypted backup of the alarms, alerts, configuration, encryption settings, schema, tasks, or trust store backends, the password used to generate the encryption key (which may have been obtained from an encryption settings definition) could have been inadvertently written into the backup descriptor. This problem does not affect local DB backends (like userRoot), the LDAP changelog backend, or the replication database.

  * When running certain command-line tools with an argument instructing the tool to read a password from a file, the password contained in that file could have been written into the server's tool invocation log instead of the path to that file. Affected tools include backup, create-initial-config, create-initial-proxy-config, dsreplication, enter-lockdown-mode, export-ldif, import-ldif, ldappasswordmodify, leave-lockdown-mode, manage-tasks, manage-topology, migrate-ldap-schema, parallel-update, prepare-endpoint-server, prepare-external-server, realtime-sync, rebuild-index, re-encode-entries, reload-http-connection-handler-certificates, reload-index, remove-defunct-server, restore, rotate-log, and stop-server. Other tools are not affected. Also note that this only includes passwords contained in files that were provided as command-line arguments; passwords included in the tools.properties file, or in a file referenced from tools.properties, would not have been exposed.

  In each of these cases, the files would have been written with permissions that make their contents only accessible to the system account used to run the server. Further, while administrative passwords may have been exposed in the tool invocation log, neither the passwords for regular users, nor any other

data from their entries, should have been affected. We have introduced new automated tests to help ensure that such incidents do not occur in the future.

We recommend changing any administrative passwords you fear may have been compromised as a result of this issue. If you are concerned that the passphrase for an encryption settings definition may have been exposed, then we recommend creating a new encryption settings definition that is preferred for all subsequent encryption operations, exporting your data to LDIF, and re-importing so that it will be encrypted with the new key. You also may wish to re-encrypt or destroy any existing backups, LDIF exports, or other data encrypted with a compromised key, and you may wish to sanitize or destroy any existing tool invocation log files that may contain clear-text passwords.

- Fixed in: 7.3.0.0
- Introduced in: 7.0.0.0
- Support identifiers: DS-38897 DS-38908

- The following enhancements were made to the topology manager to make it easier to diagnose the connection errors:

  - Added monitoring information for all the failed outbound connections (including the time since it's been failing and the last error message seen when the failure occurred) from a server to one of its configured peers and the number of failed outbound connections.

  - Added alarms/alerts for when a server fails to connect to a peer server within a configured grace period.

  - Fixed in: 7.2.1.0
  - Introduced in: 7.0.0.0
  - Support identifiers: DS-38334 SF#00655578
- The topology manager will now raise a mirrored-subtree-manager-connection-asymmetry alarm when a server is able to establish outbound connections to its peer servers, but those peer servers are unable to establish connections back to the server within the configured grace period. The alarm is cleared when connection symmetry is achieved.

  - Fixed in: 7.2.1.0
  - Introduced in: 7.0.0.0
  - Support identifiers: DS-38344 SF#00655578
- The dsreplication tool has been fixed to work when the node being used to enable replication is currently out-of-sync with the topology master.

  - Fixed in: 7.2.1.0
  - Introduced in: 7.0.0.0
  - Support identifiers: DS-38335 SF#00655578
- Fixed two issues in which the server could have exposed some clear-text passwords in files on the server file system.

  * When creating an encrypted backup of the alarms, alerts, configuration, encryption settings, schema, tasks, or trust store backends, the password used to generate the encryption key (which may have been obtained from an encryption settings definition) could have been inadvertently written into the backup descriptor. This problem does not affect local DB backends (like userRoot), the LDAP changelog backend, or the replication database.

  * When running certain command-line tools with an argument instructing the tool to read a password from a file, the password contained in that file could have been written into the server's tool invocation log instead of the path to that file. Affected tools include backup, create-initial-config, create-initial-proxy-config, dsreplication, enter-lockdown-mode, export-ldif, import-ldif, ldappasswordmodify, leave-lockdown-mode, manage-tasks, manage-topology, migrate-ldap-schema, parallel-update, prepare-endpoint-server, prepare-external-server, realtime-sync, rebuild-index, re-encode-entries, reload-http-connection-handler-certificates, reload-index, remove-defunct-server, restore, rotate-log, and stop-server. Other tools are not affected. Also note that this only includes passwords contained in files that

were provided as command-line arguments; passwords included in the tools.properties file, or in a file referenced from tools.properties, would not have been exposed.

In each of these cases, the files would have been written with permissions that make their contents only accessible to the system account used to run the server. Further, while administrative passwords may have been exposed in the tool invocation log, neither the passwords for regular users, nor any other data from their entries, should have been affected. We have introduced new automated tests to help ensure that such incidents do not occur in the future.

We recommend changing any administrative passwords you fear may have been compromised as a result of this issue. If you are concerned that the passphrase for an encryption settings definition may have been exposed, then we recommend creating a new encryption settings definition that is preferred for all subsequent encryption operations, exporting your data to LDIF, and re-importing so that it will be encrypted with the new key. You also may wish to re-encrypt or destroy any existing backups, LDIF exports, or other data encrypted with a compromised key, and you may wish to sanitize or destroy any existing tool invocation log files that may contain clear-text passwords.

- Fixed in: 7.0.1.3
- Introduced in: 7.0.0.0
- Support identifiers: DS-38897 DS-38908

Upgrade Considerations

Important considerations for upgrading to this version of the Data Sync Server:

- If you have upgraded a server that is in a cluster (i.e., has a cluster name set in the Server Instance configuration object) to version 8.1, you will not be able to make cluster configuration changes until all servers with the same cluster name have been upgraded to version 8.1. If needed, you could create temporary clusters based on server versions and modify each of the servers' cluster name appropriately to minimize the impact while you are upgrading.
- Updated the "delay bind response" failure lockout action to provide an option to delay the response to bind requests initiated by non-LDAP clients (for example, when using HTTP basic authentication). This option is disabled by default because delaying the bind response for non-LDAP clients may require temporarily blocking the thread used to process the request, which could increase the risk of a denial-of-service attack. To help mitigate this risk, if you enable delayed bind responses for non LDAP clients, we recommend that you also increase the number of request handler threads for all enabled HTTP connection handlers.
- Updated setup to create a second encryption settings definition if data encryption is enabled. It will continue to create a definition for 128-bit AES encryption for use as the preferred definition to preserve backward compatibility with existing servers in the topology, but it will now also generate a definition for 256-bit AES encryption. The 256-bit AES definition may become the preferred definition in a future release, but you can use it now by first ensuring that any existing instances are updated to contain the new definition (with the "encryption-settings export" and "encryption-settings import" commands) and then making it the preferred definition (with "encryption-settings set-preferred") in all instances.

What's New

These are new features for this release of the Data Sync Server

- In an ongoing effort to improve the use of containers for PingDirectory, several features have been implemented:

  - The --outputFile option has been added to the collect-support-data tool. You can now specify either a path, a file name, or a path and file name for the resulting CSD file. This means an administrator can

run the collect-support-data tool and put the output file into a directory outside of the container, allowing access to the file without having to actually connect to the container.

- The collect-support-data tool can now be run as a recurring task. Recurring tasks can be created using the Administration console which means that administrators do not have to connect to the container in order to run the tool.

- A Collect Support Tool Extended Operation has been added allowing LDAP clients to initiate the collect-support-data tool and to receive the output of the request. The LDAP SDK has been updated to support this, and the --remoteServer added to the collect-support-data tool can be used to send the request to another server. In other words, you can now run collect-support-data on the command line and reference another server, possibly in a container, and retrieve the output file remotely.

- PingDirectory has a Consent REST API that allows users to create and store consents. A new feature now allows users to search for consents that have been granted to them by another party
- Updated the server to use /dev/urandom (on non-Windows systems where that path exists and is readable) instead of /dev/random as the primary source for secure random data. Attempts to read from /dev/random can block if the underlying system does not have sufficient entropy, which can have a severe adverse effect on performance. Reads from /dev/urandom will not block, and the data that it provides is no less secure than data from /dev/random in any way that matters for the server.
- Replaced the ldapcompare tool with a new version that offers more functionality, including support for multiple compare assertions, following referrals, additional controls, and multiple output formats (including tab-delimited text, CSV, and JSON).
- Replaced the ldifsearch, ldifmodify, and ldif-diff command-line tools with more full-featured and robust implementations.
- Updated the server to require a minimum key size of 2048 bits when negotiating a TLS cipher suite that uses ephemeral Diffie-Hellman key exchange.
- Replaced the ldappasswordmodify tool with a new version that offers more functionality, including support for additional controls, support for multiple password change methods (the password modify extended operation, a regular LDAP modify operation, or an Active Directory-specific modify operation), and the ability to generate the new password on the client.
- Added a config/sample-dsconfig-batch-files directory with set of well commented dsconfig batch files that may be useful in enabling or configuring a variety of features in the server.

Known Issues/Workarounds

The following are known issues in the current version of the Data Sync Server

- Several known issues can occur when you use the Administrative Console with Tomcat 9.0.31. You can resolve these issues by upgrading to Tomcat 9.0.33 or later.
- If you use the create-systemd-script tool to create a forking systemd service, the service is stopped by the "systemctl stop ping-directory.service" command. At that time, you can see the status using the "systemctl status ping-directory.service" command. That status might contain an indication of failure: "Active: failed (Result: exit-code)". This error has to do with the way the service exits. It is harmless.

Resolved Issues

The following issues have been resolved with this release of the Data Sync Server:

Ticket ID	Description
DS-1046,DS-1204,DS-36547	Added support for remotely invoking the collect-support-data tool using an administrative task, and for invoking the tool on a regular basis as a recurring task. The tool has also been updated to add an outputPath argument to allow specifying the path or name to use for the output file.

Ticket ID	Description
DS-37829	The "create-systemd-script" CLI now creates a "forking" service file since Ping services are started by a process (the "start-server" script) that is different than the actual service process.
DS-38122	Added support for an extended operation that can be used to invoke the collect-support-data tool from a remote system and stream the output and resulting support data archive back to the client. The collect-support-data command-line tool has been updated to support this capability through the new --useRemoteServer argument.
DS-38535	Fixed an issue that could cause the server to generate an administrative alert about an uncaught exception when trying to send data on a TLS-encrypted connection that is no longer valid.
DS-39798	Fixed a bug in which SEMI_AGGRESSIVE and AGGRESSIVE JVM Tuning Parameters were previously allowed to both be selected.
DS-39862	Improved support for the PingOne Sync Source and Destination in the create-sync-pipe-config tool by adding Sync Classes tailored to the PingOne endpoint.
DS-40025,DS-42041	When using an Active Directory Sync Source, the base DN of the Sync Source can now be set to a descendant of the root directory partition of the AD instance(s). When detecting changes, the Sync Source will adjust the value to be the root directory partition if it is not already set to this value, but when synchronizing changes, the Sync Source will only consider changes that have a DN under the base DN provided to the Sync Source.
DS-40241	Updated create-sync-pipe-config to support creating a sync pipe that does not synchronize any attributes. This will be suggested as the default option when either the source or destination is Active Directory or a relational database. The previous default of synchronizing all attributes will still be used when both the source and destination are non-Active Directory LDAP servers.
DS-40356	Updated the manage-profile tool to prevent displaying warnings about offline config changes when starting the server.

Ticket ID	Description
DS-40413	Updated the behavior of the resync tool when an invalid DN is detected. Instead of stopping the process, the offending entry will be reported and skipped, allowing for the remaining entries to be processed. This issue only affected ActiveDirectory and Changelog Sync Sources (LDAP Sync Sources that detect changes using cn=changelog, such as PingDirectory or ODSEE).
DS-40532	Added a logging-error-behavior property to the log publisher, periodic stats logger plugin, and monitor history plugin configuration that can be used to specify the behavior the server should exhibit if an error occurs while attempting logging-related processing. By default, the server will preserve its previous behavior of writing a message to standard error, but it can be configured to enter lockdown mode on a logging error, in which the server will report itself as unavailable and will only accept requests from accounts with the lockdown-mode privilege and only from clients communicating over a loopback interface.
DS-40551	Fixed an issue that could prevent some tools from running properly with an encrypted tools.properties file.
DS-40567	A license is now always required when using the manage-profile replace-profile tool.
DS-40746	Updated the logic that the server uses to select an appropriate default set of TLS cipher suites.
DS-40806	Fixed an issue that could cause the shutdown process to stall if the server is configured to use TCP to communicate with a StatsD endpoint that has become unresponsive.
DS-40889	Fixed an issue with recurring exec tasks where the working-directory attribute was ignored.
DS-41054	Fixed an issue that stopped new extensions from being installed.
DS-41074	Fixed an issue with the way the server reports memory usage after completing an explicitly requested garbage collection.
DS-41086	Updated the StatsD monitoring endpoint to replace any spaces, commas, or colons with underscores, and remove and single quotes or double quotes in sent metric lines. This simplifies parsing of the produced metrics.

Ticket ID	Description
DS-41118	A gauge called HTTP Processing (Percent) is now available. This gauge measures the server's capacity to process new incoming HTTP requests.
DS-41126	Updated the server to make the general monitor entry available to JMX clients.
DS-41142	Improved debugging support for Server SDK extensions. If debugging is enabled, the server will now generate a debug message whenever it invokes an extension. For some extension methods that return a value, the server will also generate a debug message with that return value.
DS-41206	Fixed a memory leak when performing SCIM queries on the Directory Server.
DS-41235	Updated the cn=Cluster subtree to prevent clustered configuration changes when servers in the cluster have mixed versions. To make clustered configuration changes, either update all servers in the cluster to the same version, or temporarily create separate clusters by server version by changing the cluster-name property on the server instance configuration objects.
DS-41236	To avoid inconsistencies, changing clustered configuration will now require all servers in the cluster to be on the same product version. Servers will not pull any clustered configuration from the master of the cluster if they are on a different product version.
DS-41261	Fixed an issue with manage-profile replace-profile where certain configuration changes for recurring task chains were not being applied.
DS-41263	Updated the README for the Ping Identity Active Directory Password Sync Agent to indicate support for Windows Server 2016 and Windows Server 2019 and the removal of support for Windows Server 2008.
DS-41289	Fixed an issue that prevented password changes for topology administrators unless their password policy was configured to allow pre-encoded passwords.
DS-41301	Addressed an issue that could lead to slow, off-heap memory growth. This only occurred on servers whose cn=Version,cn=monitor entry was retrieved frequently.

Ticket ID	Description
DS-41333	Added an ssl-client-auth-policy configuration property to the HTTP connection handler to provide support for mutual TLS authentication.
DS-41366	Updated the base monitor entry to include locationName and locationDN attributes if the server is configured with a location.
DS-41396	Updated the Server SDK to add ClientContext and OperationContext methods for obtaining the name and DN of the associated client connection policy.
DS-41400	Updated the file servlet HTTP servlet extension to add support for requiring authentication in order to access the content. Access may optionally be limited to members of a specified set of groups.
DS-41579	Fixes issue with reatime-sync Password Files and Pin Files displaying contents in clear text.
DS-41731	Fixed an issue that could prevent setup from generating a self-signed certificate for systems with non-ASCII hostnames.
DS-41762	Fixed an issue where mirrored subtree polling could produce config archive files that were identical or ignored the configured insignificant attributes list.
DS-41784	Fixed a bug that could cause the duration of a sync operation to be miscalculated.
DS-41818	Added the --zip argument to the manage-profile generate-profile subcommand, which can be used to generate a zipped server profile.
DS-41820	Added an administrative task that may be used to generate a server profile and a corresponding recurring task that may be used to invoke the task on a regular basis.
DS-41821	Added an instance root file servlet to the default configuration. HTTPS requests to /instance-root by authenticated users with the file-servlet-access privilege will be granted access to files within the server instance root.
DS-41850	Servers running on Linux will now log a warning about possible performance impacts if the current memory control group has memory.swappiness set to a nonzero value.

Ticket ID	Description
DS-42006	The server now warns the administrator at startup if there are multiple versions of the same jar listed in the classpath, and the first one in the classpath is not the newest one.
DS-42033	Addressed an issue where some tools would throw a NullPointerException if a server was configured with a custom global result code map.
DS-42117	Updated Constructed Attribute Mapping's exclude-value property to accept multiple values with different capitalizations.
DS-42387	Updated the **`manage-profile generate-profile`** subcommand to exclude files in the `ldif/` and `bak/` directories by default when generating a server profile. If necessary, you can manually include those directories using the `--includePath` argument.

PingDataSync Server 8.0.0.5 release notes

Critical fixes

This release of the Data Sync Server addresses critical issues from earlier versions. Update all affected servers appropriately.

No critical issues have been identified.

PingDataSync 8.0.0.3 release notes

Critical Fixes

This release of PingDataSync addresses critical issues from earlier versions. Update all affected servers appropriately.

No critical issues have been identified

Resolved Issues

The following issues have been resolved with this release of PingDataSync:

Ticket ID	Description
DS-38535	Fixed an issue that could cause the server to generate an administrative alert about an uncaught exception when trying to send data on a TLS-encrypted connection that is no longer valid.
DS-41807	Fixed an issue where LDAP DELETE operations were sometimes not synchronized from an Oracle Unified Directory sync source, due to variations in the format of the 'targetUniqueID' attribute obtained from the Oracle changelog.

Ticket ID	Description
DS-43288	Updated setup and the replace-certificate tool to improve the way we generate self-signed certificates and certificate signing requests to make them more palatable to clients. To reduce the frequency with which administrators had to replace self-signed certificates, we previously used a very long lifetime for self-signed certificates generated by setup or the replace-certificate tool. However, some clients (especially web browsers and other HTTP clients) have started more strenuously objecting to certificates to long lifetimes, so we now generate self-signed certificates with a one-year validity period. The inter-server certificate (which is used internally within the server and does not get exposed to normal clients) is still created with a twenty-year lifetime. Also, the replace-certificate tool's interactive mode has been updated to improve the process that it uses to obtain information to include in the subject DN and subject alternative name extension for self-signed certificates and certificate signing requests. The following changes have been made in accordance with CA/Browser Forum guidelines: * When selecting the subject DN for the certificate, we listed a number of common attributes that may be used, including CN, OU, O, L, ST, and C. We previously indicated that CN attribute was recommended. We now also indicate that the O and C attributes are recommended as well. * When obtaining the list of DNS names to include in the subject alternative name extension, we previously suggested all names that we could find associated with interfaces on the local system. In many cases, we now omit non-qualified names and names that are associated with loopback interfaces. We will also warn about any attempts to add unqualified or invalid names to the list. * When obtaining the list of IP addresses to include in the subject alternative name extension, we previously suggested all addresses associated with all network interfaces on the system. We no longer suggest any IP addresses associated with loopback interfaces, and we no longer suggest any IP addresses associated in IANA-reserved ranges (for example, addresses reserved for private-use networks). The tool will now warn about attempts to add these addresses for inclusion in the subject alternative name extension.

Ticket ID	Description
DS-43480	Updated the system information monitor provider to restrict the set of environment variables that may be included. Previously, the monitor entry included information about all defined environment variables, as that information can be useful for diagnostic purposes. However, some deployments may include credentials, secret keys, or other sensitive information in environment variables, and that should not be exposed in the monitor. The server will now only include values from a predefined set of environment variables that are expected to be the most useful for troubleshooting problems, and that are not expected to contain sensitive information.
DS-43632	Fixed an issue where the "format" field is omitted from the list of operational attribute schemas in the Directory REST API.
DS-43651	The Security Guide is now available online at pingidentity.com and has been removed from the server packaging.

PingDataSync 8.0.0.2 Release Notes

Critical Fixes

This release of the Data Sync Server addresses critical issues from earlier versions. Update all affected servers appropriately.

No critical issues have been identified

Resolved Issues

The following issues have been resolved with this release of the Data Sync Server:

Ticket ID	Description
DS-37890	Added a new property to sync classes called replace-all-attr-values- limit. The property is only used when replace-all-attr-values is true and the sync destination is an LDAP server. If a modification would replace more values of the same attribute than the value of replace-all- attr-values-limit, then the modification will use ADD or DELETE operations rather than a REPLACE operation.
DS-40551	Fixed an issue that could prevent some tools from running properly with an encrypted tools.properties file.

Ticket ID	Description
DS-40828	Fixed an issue where some state associated with a JMX connection was not freed after the connection was closed. This led to a slow memory leak in servers that were monitored by an application that created a new JMX connection each polling interval.
DS-41054	Fixed an issue that stopped new extensions from being installed.
DS-41074	Fixed an issue with the way the server reports memory usage after completing an explicitly requested garbage collection.
DS-41126	Updated the server to make the general monitor entry available to JMX clients.
DS-41235	Updated the cn=Cluster subtree to prevent clustered configuration changes when servers in the cluster have mixed versions. To make clustered configuration changes, either update all servers in the cluster to the same version, or temporarily create separate clusters by server version by changing the cluster-name property on the server instance configuration objects.
DS-41236	To avoid inconsistencies, changing clustered configuration will now require all servers in the cluster to be on the same product version. Servers will not pull any clustered configuration from the master of the cluster if they are on a different product version.
DS-41261	Fixed an issue with manage-profile replace-profile where certain configuration changes for recurring task chains were not being applied.
DS-41289	Fixed an issue that prevented password changes for topology administrators unless their password policy was configured to allow pre-encoded passwords.

Ticket ID	Description
DS-42609	Fixed an issue in which the Directory REST API could fail to decode certain credentials when using basic authentication.
DS-42665	Fixed an issue that caused the resync tool to not take the attribute-comparison-method of the sync class into account. This caused resync to ignore when byte-for-byte comparison was configured.
DS-42812	Upgrade to jetty 9.4.30

PingDataSync 8.0.0.1 Release Notes

Critical Fixes

This release of the Data Sync Server addresses critical issues from earlier versions. Update all affected servers appropriately.

- Fixed a memory leak when performing SCIM queries on PingDataMetrics Server.
  - Fixed in: 8.0.0.1

    Introduced in: 7.2.0.0

    Support identifiers: DS-41206 SF#00681395
- Addressed an issue that could lead to slow off-heap memory growth. This only occurred on servers whose cn=Version,cn=monitor entry was retrieved frequently.
  - Fixed in: 8.0.0.1

    Introduced in: 5.2.0.0

    Support identifiers: DS-41301
- The following enhancements were made to the topology manager to make it easier to diagnose connection errors:

  - Added monitoring information for all the failed outbound connections (including the time since it's been failing and the last error message seen when the failure occurred) from a server to one of its configured peers and the number of failed outbound connections.

  - Added alarms/alerts for when a server fails to connect to a peer server within a configured grace period.

  - Fixed in: 7.3.0.0
  - Introduced in: 7.0.0.0
  - Support identifiers: DS-38334 SF#00655578
- The topology manager will now raise a mirrored-subtree-manager-connection-asymmetry alarm when a server is able to establish outbound connections to its peer servers, but those peer servers are unable to establish connections back to the server within the configured grace period. The alarm is cleared as soon as there is connection symmetry.
  - Fixed in: 7.3.0.0
  - Introduced in: 7.0.0.0
  - Support identifiers: DS-38344 SF#00655578

- The dsreplication tool has been fixed to work when the node being used to enable replication is currently out-of-sync with the topology master.
  - Fixed in: 7.3.0.0
  - Introduced in: 7.0.0.0
  - Support identifiers: DS-38335 SF#00655578
- Fixed two issues in which the server could have exposed some clear-text passwords in files on the server file system.

  * When creating an encrypted backup of the alarms, alerts, configuration, encryption settings, schema, tasks, or trust store backends, the password used to generate the encryption key (which may have been obtained from an encryption settings definition) could have been inadvertently written into the backup descriptor. This problem does not affect local DB backends (like userRoot), the LDAP changelog backend, or the replication database.

  * When running certain command-line tools with an argument instructing the tool to read a password from a file, the password contained in that file could have been written into the server's tool invocation log instead of the path to that file. Affected tools include backup, create-initial-config, create-initial-proxy-config, dsreplication, enter-lockdown-mode, export-ldif, import-ldif, ldappasswordmodify, leave-lockdown-mode, manage-tasks, manage-topology, migrate-ldap-schema, parallel-update, prepare-endpoint-server, prepare-external-server, realtime-sync, rebuild-index, re-encode-entries, reload-http-connection-handler-certificates, reload-index, remove-defunct-server, restore, rotate-log, and stop-server. Other tools are not affected. Also note that this only includes passwords contained in files that were provided as command-line arguments; passwords included in the tools.properties file, or in a file referenced from tools.properties, would not have been exposed.

  In each of these cases, the files would have been written with permissions that make their contents only accessible to the system account used to run the server. Further, while administrative passwords may have been exposed in the tool invocation log, neither the passwords for regular users, nor any other data from their entries, should have been affected. We have introduced new automated tests to help ensure that such incidents do not occur in the future.

  We recommend changing any administrative passwords you fear may have been compromised as a result of this issue. If you are concerned that the passphrase for an encryption settings definition may have been exposed, then we recommend creating a new encryption settings definition that is preferred for all subsequent encryption operations, exporting your data to LDIF, and re-importing so that it will be encrypted with the new key. You also may wish to re-encrypt or destroy any existing backups, LDIF exports, or other data encrypted with a compromised key, and you may wish to sanitize or destroy any existing tool invocation log files that may contain clear-text passwords.
  - Fixed in: 7.3.0.0
  - Introduced in: 7.0.0.0
  - Support identifiers: DS-38897 DS-38908
- The following enhancements were made to the topology manager to make it easier to diagnose the connection errors:

  - Added monitoring information for all the failed outbound connections (including the time since it's been failing and the last error message seen when the failure occurred) from a server to one of its configured peers and the number of failed outbound connections.

  - Added alarms/alerts for when a server fails to connect to a peer server within a configured grace period.
  - Fixed in: 7.2.1.0
  - Introduced in: 7.0.0.0
  - Support identifiers: DS-38334 SF#00655578
- The topology manager will now raise a mirrored-subtree-manager-connection-asymmetry alarm when a server is able to establish outbound connections to its peer servers, but those peer servers are unable

to establish connections back to the server within the configured grace period. The alarm is cleared when connection symmetry is achieved.

  - Fixed in: 7.2.1.0
  - Introduced in: 7.0.0.0
  - Support identifiers: DS-38344 SF#00655578
- The dsreplication tool has been fixed to work when the node being used to enable replication is currently out-of-sync with the topology master.

  - Fixed in: 7.2.1.0
  - Introduced in: 7.0.0.0
  - Support identifiers: DS-38335 SF#00655578
- Fixed two issues in which the server could have exposed some clear-text passwords in files on the server file system.

  * When creating an encrypted backup of the alarms, alerts, configuration, encryption settings, schema, tasks, or trust store backends, the password used to generate the encryption key (which may have been obtained from an encryption settings definition) could have been inadvertently written into the backup descriptor. This problem does not affect local DB backends (like userRoot), the LDAP changelog backend, or the replication database.

  * When running certain command-line tools with an argument instructing the tool to read a password from a file, the password contained in that file could have been written into the server's tool invocation log instead of the path to that file. Affected tools include backup, create-initial-config, create-initial-proxy-config, dsreplication, enter-lockdown-mode, export-ldif, import-ldif, ldappasswordmodify, leave-lockdown-mode, manage-tasks, manage-topology, migrate-ldap-schema, parallel-update, prepare-endpoint-server, prepare-external-server, realtime-sync, rebuild-index, re-encode-entries, reload-http-connection-handler-certificates, reload-index, remove-defunct-server, restore, rotate-log, and stop-server. Other tools are not affected. Also note that this only includes passwords contained in files that were provided as command-line arguments; passwords included in the tools.properties file, or in a file referenced from tools.properties, would not have been exposed.

  In each of these cases, the files would have been written with permissions that make their contents only accessible to the system account used to run the server. Further, while administrative passwords may have been exposed in the tool invocation log, neither the passwords for regular users, nor any other data from their entries, should have been affected. We have introduced new automated tests to help ensure that such incidents do not occur in the future.

  We recommend changing any administrative passwords you fear may have been compromised as a result of this issue. If you are concerned that the passphrase for an encryption settings definition may have been exposed, then we recommend creating a new encryption settings definition that is preferred for all subsequent encryption operations, exporting your data to LDIF, and re-importing so that it will be encrypted with the new key. You also may wish to re-encrypt or destroy any existing backups, LDIF exports, or other data encrypted with a compromised key, and you may wish to sanitize or destroy any existing tool invocation log files that may contain clear-text passwords.

  - Fixed in: 7.0.1.3
  - Introduced in: 7.0.0.0
  - Support identifiers: DS-38897 DS-38908
- Fixed an issue with the sync connect and response timeouts being set with incorrect units of time.

  - Fixed in: 6.2.0.0
  - Introduced in: 6.0.0.0
  - Support identifiers: DS-18026 SF#00616763
- Fixed an issue with the sync connect and response timeouts being set with incorrect units of time.

  - Fixed in: 6.2.0.0
  - Introduced in: 6.0.0.0
  - Support identifiers: DS-18026 SF#00616763

- The server can now detect an "out of file handles" situation on the operating system, and shut down to prevent running in an unreliable state.
  - Fixed in: 5.1.0.0
  - Introduced in: 2.1.0.0
  - Support identifiers: DS-12579 SF#2655
- Disabled support for SSLv3 by default in the LDAP, HTTP, and JMX connection handlers, and for replication communication. The recently-discovered POODLE vulnerability could potentially allow a network attacker to determine the plaintext behind an SSLv3-encrypted session, which would effectively negate the primary benefit of the encryption.

  SSLv3 was initially defined in 1996, but was supplanted by the release of the TLSv1 definition in 1999 (and subsequently by TLSv1.1 in 2006 and TLSv1.2 in 2008). These newer TLS protocols are not susceptible to the POODLE vulnerability, and the server has supported them (and preferred them over SSLv3) for many years. The act of disabling SSLv3 by default should not have any adverse effect on clients that support any of the newer TLS protocols. However, if there are any legacy client applications that attempt to communicate securely but do not support the newer TLS protocols, they should be updated to support the newer protocols. In the event that there are known clients that do not support any security protocol newer than SSLv3 and that cannot be immediately updated to support a newer protocol, SSLv3 support can be re-enabled using the newly-introduced allowed-insecure-tls-protocol global configuration property. However, since communication using SSLv3 can no longer be considered secure, it is strongly recommended that every effort be made to update all known clients still using SSLv3.

  It is possible to use the server access log to identify LDAP clients that use SSLv3 to communicate with the server. Whenever an LDAP client establishes a secure connection to the server, or whenever a client uses the StartTLS extended operation to secure an existing plaintext connection, the server will generate a SECURITY-NEGOTIATION access log message. The "protocol" element of a SECURITY-NEGOTIATION access log message specifies the name of the security protocol that has been negotiated between the client and the server, and any SECURITY-NEGOTIATION messages with a protocol of "SSLv3" suggest that the associated client is vulnerable to the POODLE attack. In addition, if any connections are terminated for attempting to use the disallowed SSLv3 protocol, the access log message for that disconnect should include a message stating the reason for the termination.
  - Fixed in: 5.0.0.0
  - Introduced in: 2.1.0.0
  - Support identifiers: DS-11782
- Change the default behavior of the Synchronization Server to not lock entries across all Sync Pipes when processing changes.

  The Sync Server has a specialized mutex that ensures that changes to the same entry are processed serially. The primary reason for this mutex is to ensure that the server can safely process changes in parallel to achieve high throughput. However, we also use this mutex to ensure that two Sync Pipes don't process the same entry at the same time for deployments that synchronize changes bi-directionally. A consequence of this locking is that if one Sync Pipe is failing (because the destination is unavailable) then it retains the lock on an entry, and when other Sync Pipes try to process changes to that entry they will block that change and all changes that follow it while they wait on the lock.

  This change turns off using a shared mutex by default, but adds a new advanced configuration option on the Sync Pipe, shared-mutex-name, that specifies the name of a mutex that is shared by other Sync Pipes. This gives greater control over the locking so that two Sync Pipes that share end points can

ensure that two changes to the same logical user are not processed concurrently, while not impacting other Sync Pipes.

See the shared-mutex-name property for more information.

This property is subject to change in a future release.

- Fixed in: 3.2.0.0
- Introduced in: 3.0.0.0
- Support identifiers: DS-4202 SF#1527

Resolved Issues

The following issues have been resolved with this release of the Data Sync Server:

Ticket ID	Description
DS-40532	Added a logging-error-behavior property to the log publisher, periodic stats logger plugin, and monitor history plugin configuration that can be used to specify the behavior the server should exhibit if an error occurs while attempting logging-related processing. By default, the server will preserve its previous behavior of writing a message to standard error, but it can be configured to enter lockdown mode on a logging error, in which the server will report itself as unavailable and will only accept requests from accounts with the lockdown-mode privilege and only from clients communicating over a loopback interface.
DS-41206	Fixed a memory leak when performing SCIM queries on the Directory Server.

PingDataSync 8.0.0.0 Release Notes

Critical Fixes

This release of the PingDataSync Server addresses critical issues from earlier versions. Update all affected servers appropriately.

No critical issues have been identified

What's New

These are new features for this release of the PingDataSync Server:

- Use Server Profiles to reduce risk and improve consistency following the DevOps principle of infrastructure-as-code. Administrators can export the configuration of the PingDataSync Server server to a directory of text files called a Server Profile, track changes to these files in version control such as Git, and install new instances of PingDataSync Server or update existing instances of PingDataSync Server from a Server Profile. Server Profiles support variable substitution in order to remove the settings unique to each pre-production or production environment from the Server Profile that is stored in version control.
- Improved consistency for user passwords that are managed in PingOne for Customers. Now when PingDataSync Server recognizes a password change in the source PingOne for Customer environment, the password is removed from the destination PingDirectory. Upon the next on-prem LDAP bind, the validated password is saved in the destination PingDirectory by the pass-through authentication plugin.

Known Issues/Workarounds

The following are known issues in the current version of the PingDataSync Server:

- The following are suggested solutions for problems with slow DNS:
  - Maintain a connection pool in the client app rather than opening new connections for each bind.
  - Add appropriate records, including PTR records, to DNS.
  - Add `options timeout:1` in the `/etc/resolv.conf` file and/or `options single-request`
  - If IPv6 requests specifically are causing issues, add `-Djava.net.preferIPv4Stack=true` to the `start-server.java-args` line in PingDirectory's `config/java.properties` file, run **`bin/dsjavaproperties`**, and restart the server to stop the issuance of IPv6 PTR requests.
- Some server tools, such as `dsreplication`, `collect-support-data`, and `rebuild-index`, will fail with errors if they are run with an encrypted `tools.properties` file.

  Workaround: Add the `--noPropertiesFile` argument to the server tools to prevent them from pulling information from the encrypted file.
- Deploying the Admin Console to an external container using JDK 11 requires downloading the following dependencies and making them available at runtime (for example, by copying them to the `WEB-INF/lib` directory of the exploded WAR file).
  - groupId:jakarta.xml.bind, artifactId:jakarta.xml.bind-api, version:2.3.2
  - groupId:org.glassfish.jaxb, artifactId:jaxb-runtime, version:2.3.2

  Workaround: Deploy the Console in an external container using JDK 8.

Resolved Issues

The following issues have been resolved with this release of the Data Sync Server:

Ticket ID	Description
DS-17278	Added a `cn=Server Status Timeline,cn=monitor` monitor entry to track a history of the local server's last 100 status changes and their timestamps. Updated the LDAP external server monitor to include attributes tracking health check state changes for external servers. The new attributes include the number of times a health check transition has occurred, timestamps of the most recent transitions, and messages associated with the most recent transitions.
DS-37881	The PingFederate Access Token Validator will now refresh its cached value of the PingFederate server's token introspection endpoint. A new attribute, endpoint-cache-refresh, has been added to the PingFederate Access Token Validator, which will determine how often this refresh occurs.
DS-37955	To support multiple trace loggers, each trace logger now has its own resource key, which is shown in the `Resource` column in the output of `status`. This key allows multiple alarms, due to sensitive message types for multiple trace loggers.
DS-38053	The JWT Access Token Validator no longer requires a restart after a change to one of its signing certificates.

Ticket ID	Description
DS-38493	Updated the PingOne for Customer Sync Source to remove a user's password from a destination (for example, PingDirectory) when the corresponding user's password has changed in PingOne for Customers. This behavior requires the `password` attribute in PingOne for Customers to be mapped to the password attribute (for example, `userPassword`) in the destination schema.
DS-38560	Updated `manage-profile replace-profile` to apply configuration changes directly, when possible. If the new server profile used by `replace-profile` has changed only the `dsconfig` batch files from the original profile, then only the `dsconfig` files are applied. If no changes are detected between profiles, replace-profile takes no action. If changes other than dsconfig are detected, the full replace-profile process is followed.
DS-38777	Added support for updating the server version during `manage-profile replace-profile`. The server must have been originally set up with a server profile.
DS-38832	Fixed an issue that could cause the server to leak a small amount of memory each time it failed to establish an LDAP connection to another server.
DS-38863	Updated the manage-profile setup subcommand to set a server's cluster name to match its instance name by default. This prevents servers in the same replication topology from being in the same cluster, reducing the risk of unintentionally overwriting parts of an existing server's configuration in a DevOps environment. The `--useDefaultClusterName` argument can be used to leave the cluster name unchanged.
DS-38867	Updated the PBKDF2 password storage scheme to add support for variants that use the 256-bit, 384-bit, and 512-bit SHA-2 digest algorithms. At present, the SHA-1 variant remains the default to preserve backward compatibility with older versions. Also, in accordance with the recommendations in NIST SP 800-63B, we have increased the default iteration count from 4096 to 10,000, and the default salt length from 64 bits to 128 bits.
DS-38869	Updated the `remove-defunct-server` tool's `--ignoreOnline` option. When using `--ignoreOnline` in a mixed-version environment, all servers must support the option.
DS-39176, DS-39308	Updated the Groovy scripting language version to 2.5.7. For a list of changes, visit *groovy-lang.org* and view the Groovy 2.5 release notes. As of this release, only the core Groovy runtime and the groovy-json module are bundled with the server. To deploy a Groovy-scripted Server SDK extension that requires a Groovy module not bundled with the server, such as groovy-xml or groovy-sql, download the appropriate jar file from *groovy-lang.org* and place it in the server's `lib/extensions` directory.

Ticket ID	Description
DS-39191	Updated the `auto-mapped-source-attribute` property on Sync Class configuration objects to not have a default value since the previous default of `-all-` caused confusion. Existing dsconfig batch files used for automated deployments might need to be updated to provide a value for this property. A value of `-all-` is recommended when synchronizing between two systems with the same schema, such as when migrating from a legacy LDAP server to PingDirectory.
DS-39253	Added a `replace-certificate` tool, which can help an administrator replace the listener or inter-server certificate for a server instance.
DS-39320	Added support for PingDataSync Server to the `manage-profile` tool and its subcommands.
DS-39325	Removed the legacy product-specific scripts for starting and stopping the server. These include: ▪ start-ds and stop-ds for Directory Server ▪ start-proxy and stop-proxy for Directory Proxy Server ▪ start-sync and stop-sync for Data Sync Server ▪ start-metrics-engine and stop-metrics-engine for Data Metrics Server These legacy scripts had been deprecated for several releases in favor of the more general `start-server` and `stop-server` scripts, and they displayed a warning message about their upcoming removal if they were invoked. If you still have dependencies on these legacy product-specific scripts, you will need to update them to reference the general `start-server` and `stop-server` scripts instead. If it is not feasible to update these references immediately, you may create symbolic links that use the legacy script names and point to the `start-server` and `stop-server scripts`.
DS-39373	Preserve the privileges that are explicitly set on the admin user when migrating from the admin backend to the topology registry.
DS-39518	Fixed an issue in which escaped characters in schema extensions may not be handled properly. If used in attribute type constraints (such as `X-VALUE-REGEX`), this could cause unexpected or incorrect behavior.
DS-39540	Released an updated Password Sync Agent for Active Directory that uses SSHA256 as a default for its password hashing algorithm instead of SSHA, since SSHA is soon to be deprecated in PingDirectory. If SSHA256 is not supported by a directory or if it is not wanted, a registry value was added that can be set to specify the password hashing algorithm.
DS-39592	HTTP External Servers have a new attribute, ssl-cert-nickname, which defines the alias of a specific certificate within their keystore to be used as a client certificate.
DS-39603	Fixed an issue where Server SDK extensions could not be configured by dsconfig batch files in the manage-profile tool.
DS-39626, DS-40357	The trace log publisher will now record an access token's scopes after the token is successfully validated.

Ticket ID	Description
DS-39654	Added support for the `--topologyFilePath` argument to the `manage-topology add-server` subcommand.
DS-39671	Updated the `manage-topology add-server` subcommand to require being run from the older server in a mixed-version environment.
DS-39715	Updated the Server SDK to add support for sending email messages.
DS-39853	Added support for the PingOne for Customers Sync Source and Destination to the `create-sync-pipe-config` tool.
DS-39857	Added the StatsD monitoring endpoint. When the Stats Collector Plugin is enabled, this endpoint sends metric data from the server in StatsD format to the configured destination.
DS-39908	Added a new JVM-default trust manager provider that can be used to automatically trust any certificate signed by an authority included in the JVM's default set of trusted issuers. Also, updated other trust manager providers to offer an option to use the JVM-default trust addition to the trust that they normally provide.
DS-40020	Fixed an issue in Active Directory Sync Source where the persistent state was updated before applying changes, so changes could be missed when stopping the Sync Pipe.
DS-40114	Added a new `cn=Status Health Summary,cn=monitor` monitor entry that provides a summary of the server's current assessment of its health. This simplifies monitoring with third party tools that support retrieving monitoring data over JMX. The Periodic Stats Logger has also been updated to allow some of this monitoring information to be logged. No new information is logged by default.
DS-40354	Fixed a problem with `config-diff` when writing properties that span multiple lines using the `--prettyPrint` argument.
DS-40366	Fixed an issue where the server was attempting to connect by an IP address rather than a hostname when DNS lookup was successful.
DS-40377	Added support for logging to a JSON file in the Periodic Stats Logger Plugin.
DS-40517	Added metrics for status summary, replication database, and LDAP changelog to the Stats Collector Plugin.
DS-40543	Updated `manage-profile replace-profile` to copy the tool log file to the server being updated.
DS-40556	Added support for specifying a working directory for exec tasks.
DS-40730	Updated the `encrypt-file` tool to prevent using the same path for both the input file and the output file.
DS-40771	Added a `--duration` argument to collect-support-data. When used, only the log files covering the specified duration before the current time will be collected.

PingDirectory Server Administration Guide

PingDirectory™ Product Documentation

Support

https://support.pingidentity.com/

Introduction to PingDirectory Server

PingDirectory Server is a high-performance, extensible directory and PingData Platform, written completely in Java.

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

The following topics provide an overview of the Directory Server's features and components:

Server features

The Directory Server provides an extensive feature-rich set of tools that can meet the production needs of your system.

Full LDAP version 3 implementation

The Directory Server fully supports LDAPv3, which supports the Request For Comments (RFCs) specified in the protocol. The Directory Server provides a feature-rich solution that supports the core LDAPv3 protocol in addition to server-specific controls and extended operations.

High availability

The Directory Server supports N-way multi-master replication that eliminates single points of failure and ensures high availability for a networked topology. The Directory Server allows data to be stored across multiple machines and disk partitions for fast replication. The Directory Server supports replication in entry-balancing proxy server deployments.

Administration tools

The Directory Server provides a full set of command-line tools, an administrative console, and a Java-based setup tool to configure, monitor, and manage any part of the server. The Directory Server has a task-based subsystem that provides automated scheduling of basic functions, such as backups, restores, imports, exports, restarts, and shutdowns. The set of utilities includes a troubleshooting support tool that aggregates system metrics into a `.zip` file, which administrators can send to your authorized support provider for analysis.

Self-service account manager application

The Self Service Account Manager project, hosted at *https://github.com/pingidentity/ssam*, is a customizable web application allowing users to perform their own account registration, profile updates, and password changes. The project is for testing and development purposes only and is not a supported application.

Delegated admin application

A Javascript-based web application can be installed for business users to manage identities stored in the Directory Server. The application provides delegated administration of identities for help desk or customer service representatives initiating a password reset and unlock, an employee in HR updating an address stored within another employee profile, or an application administrator updating identity attributes or group membership to allow application single sign-on (SSO) access.

Security mechanisms

The Directory Server provides extensive security mechanisms to protect data and prevent unauthorized access. Access control list (ACL) instructions are available down to the attribute value level and can be stored within each entry. The Directory Server allows connections over SSL through an encrypted communication tunnel. Clients can also use the StartTLS extended operation over standard, non-encrypted ports. Other security features include a privilege subsystem for fine-grained granting of rights, a password policy subsystem that allows configurable password validators and storage schemes, and SASL authentication mechanisms to secure data integrity, such as PLAIN, ANONYMOUS, EXTERNAL, CRAM-MD5, Digest-MD5, and GSSAPI. The Directory Server supports various providers and mappers for certificate-based authentication in addition to the ability to encrypt specific entries or sensitive attributes. For more information, see the PingDirectory Server Security Guide.

Monitoring and notifications

The Directory Server supports monitoring entries using the PingDataMetrics Server, JConsole, SNMP, or using the Administrative Console. Administrators can track the response times for LDAP operations using a monitoring histogram as well as record performance statistics down to sub-second granularity. The Directory Server supports configurable notifications, auditing, and logging subsystems with filtered logging capabilities.

Powerful LDAP SDK

The Directory Server is based on a feature-rich LDAP SDK for Java, designed by Ping Identity. The LDAP SDK is a Java API standard that overcomes the limitations of the Java Naming and Directory Interface (JNDI) model. For example, JNDI does not address the use of LDAP controls and extended operations. The LDAP SDK for Java provides support for controls and extended operations to leverage Ping Identity's extensible architecture for their applications.

ⓘ **Note:**

For more information on the LDAP SDK for Java, go to *http://www.LDAP.com*.

System for Cross-domain Identity Management (SCIM) extension

The Directory Server provides a SCIM servlet extension to facilitate moving users to, from, and between cloud-based software as a service (SaaS) applications in a secure and fast manner.

Server SDK

Ping Identity also provides the Server SDK, which is a library of Java packages, classes, and build tools to help in-house or third-party developers create client extensions for:

- PingDirectory Server
- PingDirectoryProxy Server
- Data Sync Server

The servers have a highly extensible and scalable architecture with multiple plugin points for your customization needs. The Server SDK provides APIs to alter the behavior of each server's components without affecting its code base.

Administration framework

The Directory Server provides an administration and configuration framework capable of managing stand-alone servers, server groups, and highly-available deployments that include multiple redundant server instances.

Administrators can configure changes locally or remotely:

- On a single server
- On all servers in a server group

Each server configuration is stored as a flat file (LDIF) that can be accessed under the `cn=config` branch of the directory information tree (DIT). Administrators can tune the configuration and perform maintenance functions over LDAP using a suite of command-line tools or an administrative console (for configuration and monitoring). The Directory Server provides plugins to extend the functionality of its components.

Server tools location

The Directory Server stores a full set of command-line tools for maintaining your system in the `PingDirectory/bin` directory for UNIX or Linux machines and the `PingDirectory\bat` directory for Microsoft Windows machines.

Ping Identity distributes the Directory Server, administrative console, and LDAP SDK for Java in `.zip` format. After extracting the file, you can access the **`setup`** utility in the server root directory, located at `PingDirectory`.

Before installing the directory server, see *Prepare your environment* on page 237for important information on setting up your machines. See *Installing the Server* on page 236 for information on installing a server instance using the **`setup`** utility. You can run this utility in either interactive command-line or non-interactive command-line. For information on modifying the configuration of a server instance or a group of servers using the command-line tools and the administrative console, see *Configuring the Server* on page 298.

Installing the Server

After you have prepared your hardware and software system, you can set up the server using PingDirectory Server's installation modes.

This section details the various installation options and procedures available to the administrator.

Prepare your environment

PingDirectory Server offers a highly portable and scalable architecture that runs on multiple platforms and operating systems.

The Directory Server is specifically optimized for operating systems used in environments that process a large number of entries.

See the following topics for information on setting up your server machines for optimal processing efficiency:

- *Before You Begin*
- *System requirements*
- *Platforms*
- *Docker*
- *Java Runtime Environment*
- *Browser*
- *Installing Java*
- *Preparing the Operating System (Linux)*
- *Configuring the Number of File Descriptors*
- *File System Tuning*
- *Raising the Maximum User Processes*
- *Setting Operating System Environment Variables*
- *Install systat and pstack (Red Hat)*
- *Install dstat (SUSE)*
- *Disabling File System Swapping*
- *Omit vm.overcommit_memory*
- *Managing System Entropy*
- *Setting file system event monitoring (inotify)*
- *Tuning the IO Scheduler*
- *Running as a Non-Root User (Linux)*

Before you begin

The Directory Server requires certain software packages for the proper operation of the server.

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

> ⓘ **Tip:**
>
> Implement a Network Time Protocol (NTP) system so that multi-server environments are synchronized and timestamps are accurate.

System requirements

The following section details system requirements that are qualified and certified to be compatible with the PingDirectory Server.

Differences in operating system versions, service packs, and other platform variations are supported until the platform or other required software are suspected of causing issues.

Platforms

The following operating systems support a PingDirectory Server installation and deployment.

- Windows Server 2019
- Windows Server 2016
- Red Hat Enterprise Linux ES 8.1

- Red Hat Enterprise Linux ES 7.7
- CentOS 8.1
- CentOS 7.7
- SUSE Linux Enterprise 15 SP1
- SUSE Linux Enterprise 12 SP5
- Ubuntu 18.04 LTS
- Ubuntu 16.04 LTS
- Amazon Linux 2

> ⓘ **Attention:**
>
> This product has been tested with the default configurations of all operating system components. Customized implementations or installed third-party plugins from your organization might affect the deployment of this product.

Docker
The following Docker version and operating systems are a pre-installation requirement.

- Version: Docker 19.03.5
- Host operating system: Ubuntu 18.04 LTS
- Base image operating system: Ubuntu 18.04 LTS
- Base image operating system: Alpine Linux 3.11

For more information, view the Directory Server Docker Image on DockerHub at *pingidentity/pingdirectory* and visit the *PingIdentity DevOps documentation*.

> ⓘ **Note:**
>
> Only the Directory Server software is licensed under Ping Identity's end user license agreement, and any other software components contained within the image are licensed solely under the terms of the applicable open source/third-party license.
>
> Ping Identity accepts no responsibility for the performance of any specific virtualization software and in no way guarantees the performance or interoperability of any virtualization software with its products.

Java Runtime Environment
The following Java Development Kit versions are a pre-installation requirement.

- Oracle JDK 8 64-bit
- Oracle JDK 11 LTS 64-bit
- OpenJDK 8 64-bit
- OpenJDK 11
- Corretto 8
- Corretto 11

> ⓘ **Note:**
>
> The *Ping Identity Java Support Policy* applies to your Java Runtime Environment.

Browsers

Administration Console

- Chrome
- Firefox

- Internet Explorer 11 or later

End users

- Chrome
- Edge
- Firefox
- Internet Explorer 11 or later
- Safari

Installing Java
Even if your system already has Java installed, you might want to create a separate Java installation for use by the PingDirectory Server so that updates to the system-wide Java installation do not impact the Directory Server.

About this task

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

> ⓘ **Note:**
>
> A separate Java installation requires that you download the Java Development Kit (JDK) instead of the Java Runtime Environment (JRE) for the 64-bit version.

Steps

1. Go to the *Oracle download site*.
2. Click JDK Download.
3. On the download page, select the download file for your operating system and accept the license agreement to start the download.

Preparing the operating system (Linux)
Several operating system customizations can improve the performance of PingDirectory Server on Linux systems.

These customizations include:

- *Increasing the file descriptor limit on Linux systems*
- *Setting the file system flushes* on page 241
- *Editing OS-level environment variables*
- *Downloading some useful monitoring tools for Redhat Linux systems*

Configuring the file descriptor limits
Operating system default file descriptor limitsrestrict the number of PingDirectory Server connections. Change the descriptor limits to allow more connections.

About this task

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

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

After you set the operating system limit, you can configure the number of file descriptors that the server will use either by using a **NUM_FILE_DESCRIPTORS** environment variable, or by creating a **config/num-file-descriptors** file with a single line such as **NUM_FILE_DESCRIPTORS=12345**. If these are not set, the operating system uses the default of 65535 descriptors. This is an optional change you can make if you want to ensure that the server shuts down safely prior to reaching the file descriptor limit.

Steps

1. Display the current `fs.file-max` limit of the system.

   ```
   sysctl fs.file-max
   ```

   The `fs.file-max` limit is the maximum server-wide file limit you can set without tuning the kernel parameters in the `proc` file system.
2. Edit the `/etc/sysctl.conf` file.

   If there is a line that sets the value of the `fs.file-max` property, make sure that its value is set to at least 1.5 times the per-process limit.

   If there is no line that sets a value for this property, add the following to the end of the file:

   ```
   fs.file-max = 100000
   ```

   ⓘ **Note:**

   100000 is just an example here. Specify a value of at least 1.5 times the per-process limit.
3. Display the current hard limit of the system.

   ```
   ulimit -aH
   ```

   The `open files (-n)` value is the maximum number of open files per process limit.

   The value should be set to at least 65535.
4. Edit the `/etc/security/limits.conf` file.

   If the file has lines that set the soft and hard limits for the number of file descriptors, make sure the values are set to 65535. If the lines are not present, add the following lines before `#End of file`, making certain to insert a tab between the columns.

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

   ⓘ **Note:**

   The number of open file descriptors is limited by the physical memory available to the host. You can determine this limit with the following command.

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

   If the file-max value is significantly higher than the 65535 limit, consider increasing the file descriptor limit to between 10% and 15% of the system-wide file descriptor limit. For example, if the file-max value is 810752, you could set the file descriptor limit to 100000. If the file-max value is lower than 65535, the host is likely not sized appropriately.

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

```
# ulimit -n
```

Results

ⓘ **Note:**

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

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

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

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

Tuning the file system

Administrators can tune ext3 and ext4 file systems by setting the file system flushes and noatime to improve server performance.

Newer ext4 systems use delayed allocation to improve performance. This delays block allocation until it writes data to disk. Delayed allocation improves performance and reduces fragmentation by using the actual file size to improve block allocation.

This feature might increase the risk of data loss in cases where a system loses power before all of the data has been written to disk. This can occur if a program is replacing the contents of a file without forcing a write to the disk with **`fsync`**. To reduce this risk, make sure the default **`auto_da_alloc`** option is enabled on ext4 file systems.

The following changes can be made in the `/etc/fstab` file.

Setting the file system flushes

Improve Directory Server performance by reducing the span between file system flushes.

About this task

By default, Linux servers running the ext3 file system flush the data to disk every five seconds.

Steps

1. If the Directory Server is running on a Linux server using the ext3 file system, consider editing the mount options for that file system to include the following command.

```
commit=1
```

   This variable changes the flush frequency from five seconds to one second.
2. You should also set the flush frequency in the `/etc/fstab` file.

   Changing the mount command alone does not survive across reboots.

Setting noatime on ext3 and ext4 Systems
If your Linux server uses an ext3 or ext4 file system, set `noatime` to improve performance by turning off any *atime* updates during read access.

About this task

Additionally, set the flush frequency to the `/etc/fstab` file. Performing the change through the **`mount`** command alone does not survive across reboots.

Steps

- Run the **`mount`** command for the relevant system.
  - ext3 system:

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

    .
  - ext4 system:

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

    .

Setting the maximum user processes
To address memory problems when running multiple servers, increase the default maximum user process limit.

About this task

Some Linux distributions, such as Redhat Enterprise Linux Server and CentOS 6.0 or later, set the default maximum number of user processes to 1024. This is considerably lower than the same parameter on older distributions. The default value of 1024 leads to some Java virtual machine memory errors when running multiple servers on a machine because each Linux thread counts as a user process.

At startup, the Directory Server and its tools automatically attempt to raise the maximum user processes limit to 16,383 if the value reported by **`ulimit`** is less than that. If, for any reason, the server is unable to automatically set the maximum processes limit to 16,383, it displays an error message.

There are several ways to set the maximum user process limit.

Steps

- Set the limit in `/etc/security/limits.conf`.

  Replace the (*) with the name of the user under which the software will run.

  ```
  * soft nproc 65535
  * hard nproc 65535
  ```
- Set the *NUM_USER_PROCESSES* environment variable to `16383`.
- In the `config/num-user-processes` file, set the *NUM_USER_PROCESSES* to `16383`.

About editing OS-level environment variables
Certain environment variables might affect the Directory Server in unexpected ways. This is particularly true for environment variables that are used by the underlying operating system to control how it uses non-default libraries.

For this reason, the Directory Server explicitly overrides the values of key environment variables like <*PATH*>, <*LD_LIBRARY_PATH*>, and <*LD_PRELOAD*> to ensure that environment settings used to start the server do not inadvertently impact its behavior.

If you need to edit any of these environment variables, set the values of those variables by manually editing the <*set_environment_vars*> function of the `lib/_script-util.sh` script.

You must stop (bin/stop-server) and re-start (bin/start-server) the server for the change to take effect.

Installing sysstat and pstack (Red Hat)
If you plan to run PingDirectory on a Red Hat Linux system, install a couple of packages, **`sysstat`** and **`pstack`**, that are disabled by default, but are useful for troubleshooting.

About this task

The troubleshooting tool **`collect-support-data`** uses the **`iostat`**, **`mpstat`**, and **`pstack`** utilities to collect monitoring, performance, and stack trace information on the server's processes.

Steps

- To install **`systat`** on your Red Hat system, run the following command.

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

.

Installing dstat (SUSE Linux)
`dstat` is a system monitoring tool that can help you troubleshoot PingDirectory servers on SUSE distributions.

About this task

The **`collect-support-data`** tool uses the **`dstat`** utility for system monitoring. **`dstat`** can be obtained from the OpenSUSE project website. The following process shows how to install the **`dstat`** utility on SUSE Enterprise Linux 11 SP2:

Steps

1. Sign on as the root user.
2. Add the appropriate repository using the **`zypper`** tool.
3. Install the **`dstat`** utility.

```
$ zypper install dstat
```

Disabling file system swapping
Because file system swapping can interfere with PingDirectory, disable performance tuning tools like **`tuned`**.

About this task

Steps

1. Sign on as the root user.
2. Add the line `vm.swappiness = 0` to the file `/etc/sysctl.conf`.
3. Restart the system to apply the change.
4. Optional: If you need to tune performance after disabling file system swapping, do the following:
   a. Clone the existing performance profile.
   b. Run **`tuned`**.
   c. Add the line `vm.swappiness = 0` to the file `/usr/lib/tuned/profile-name/tuned.conf`.
   d. To select the updated profile, run **`tuned-adm profile customized_profile`**.
   e. Restart the system to apply the changes.

Omitting vm.overcommit_memory
An improperly configured value for the `vm.overcommit_memory` property in the `/etc/sysctl.conf` file might cause the **`setup`** or **`start-server`** tool to fail.

About this task

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

Steps

Omit the `vm.overcommit_memory` property in the `/etc/sysctl.conf` file to ensure that enough memory is available for these tools.

Managing system entropy
Linux uses entropy to calculate random data that is used by the system in cryptographic operations.

About this task

Some environments with low entropy might have intermittent performance issues with SSL-based communication. This is more typical on virtual machines but can occur in physical instances as well.

Steps

- Monitor the **`kernel.random.entropy_avail`** in **`sysctl`** value for best results.

Setting file system event monitoring (inotify)
An event monitoring tool such as **`inotify`** can be configured for notifying processes about file system events, including file creation, deletion, and updates.

About this task

Linux limits the number of **`inotify`** watches a user can receive.

Steps

- To increase the limit, edit **`etc/sysctl.conf`** to add the following line.

```
fs.inotify.max_user_watches = 524288
```

- Run the following command.

```
$ sudo sysctl -w fs.inotify.max_user_watches=524288
```

Tuning the I/O scheduler
Using the correct I/O scheduler can increase performance and reduce the possibility of database timeouts when the system is under extreme write load.

About this task

For file systems running on an SSD or in a virtualized environment, use the recommended **`noop`** scheduler. For all other systems, use the **`deadline`** scheduler.

The procedure for configuring a scheduler to use at startup depends on the version of Linux. See the Linux documentation for your specific version for the correct way to configure this setting.

Steps

1. To determine which scheduler is configured on your system, run the following command.

```
$ cat /sys/block/<block-device>/queue/scheduler
```

2. To change the scheduler on a running system, run the following command.

```
$ echo 'deadline' > /sys/block/sda/queue/scheduler
```

3. To apply the change, restart the system.

Running as a non-root user (Linux)
You have two options to run as a non-root user but still allow connections on a privileged port.

Use a load balancer or directory proxy server

Many environments can run the server on a non-privileged port but be hidden by a hardware load balancer or LDAP directory proxy server.

Use `netfilter`

Use the **`netfilter`** mechanism, exposed through the **`iptables`** command, to automatically redirect any requests from a privileged port to the unprivileged port on which the server is listening.

Enabling the server to listen on privileged ports (Linux)
For your convenience, enable the server to listen on privileged ports while running as a non-root user.

About this task

Linux systems have a mechanism called capabilities that is used to grant specific commands the ability to do things that are normally only allowed for a root account:

- The **`setcap`** command assigns capabilities to an application.
- The **`cap_net_bind_service`** capability enables a service to bind a socket to privileged ports (port numbers less than 1024).

Steps

1. If Java is installed in **`/ds/java`** (and the Java command to run the server is **`/ds/java/bin/java`**), you can grant the **`cap_net_bind_service`** capability to the Java binary with the following command.

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

2. Create the file **`/etc/ld.so.conf.d/libjli.conf`** with the path to the directory that contains the **`libjli.so`** file.

   The java binary needs an additional shared library (**`libjli.so`**) as part of the Java installation. Because this process imposes stricter limits on where the operating system looks for shared libraries to load for commands that have capabilities assigned, it is also necessary to tell the operating system where to look for this library.

   For example, if the Java installation is in **`/ds/java`**, the contents of that file should be:

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

3. To apply the changes, run the following command.

```
$ sudo ldconfig -v
```

Getting the installation packages

Obtain the latest `.zip` release bundle from Ping Identity and extract it to a folder of your choice to begin the installation process.

About this task

The release bundle contains the Directory Server code, tools, and package documentation.

Complete the following steps to unpack the build distribution.

Steps

1. Download the latest `.zip` distribution of the Directory Server software.
2. Extract the compressed `.zip` archive file to a directory of your choice.

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

You can set up the Directory Server.

Sign on to the Administrative Console

After the server is installed, access the Administrative Console, `https://<host>/console/login`, to verify the configuration and manage the server. The root user DN or the common name of a root user DN is required to log into the Administrative Console. For example, if the DN created when the server was installed is `cn=Directory Manager`, `directory manager`can be used to log into the Administrative Console.

If the Administrative Console needs to run in an external container, such as Tomcat, a separate package can be installed according to that container's documentation. Contact Ping Identity Customer Support for the package location and instructions.

ⓘ **Note:** The session timeout for the console is 24 hours by default. When this duration is exceeded, all inactive users are logged off automatically.

To set a different timeout value, configure the `server.sessionTimeout` application parameter, which specifies the timeout duration in seconds. You can set the value as an init parameter either in the console or on the command line, as shown below.

- Console

  Select **Web Application Extensions**# **Console** and then use the **Init Parameter** field.
- Command line

  The following example uses a value of 1800 seconds (30 minutes).

```
dsconfig set-web-application-extension-prop --no-prompt \
--extension-name Console \
--add init-parameter:server.sessionTimeout=1800
```

For the change to take effect, restart the HTTP(S) Connection Handler or the server itself.

Directory Server folder layout

After extracting the Directory Server distribution file, you see the following folders and command-line utilities.

Layout of the Directory Server Folders

Directories/Files/Tools	Description
`License.txt`	Licensing agreement for the Directory Server.
`README`	`README` file that describes the steps to set up and start the Directory Server.
`bak`	Stores the physical backup files used with the **`backup`** command-line tool.
`bat`	Stores Windows-based command-line tools for the Directory Server.
`bin`	Stores UNIX/Linux-based command-line tools for the Directory Server.
`classes`	Stores any external classes for server extensions.
`collector`	Used by the server to make monitored statistics available to PingDataMetrics Server.
`config`	Stores the configuration files for the backends (admin, config) as well as the directories for messages, schema, tools, and updates .
`db`	Stores the Oracle Berkeley Java Edition database files for the Directory Server.
`docs`	Provides the product documentation.
`import-tmp`	Stores temporary imported items.
`ldif`	Stores any LDIF files that you might have created or imported.
`legal-notices`	Stores any legal notices for dependent software used with the Directory Server.
`lib`	Stores any scripts, jar, and library files needed for the server and its extensions.
`locks`	Stores any lock files in the backends.
`logs`	Stores log files for the Directory Server.
`metrics`	Stores the metrics that can be gathered for this server and surfaced in PingDataMetrics Server.
`resource`	Stores the MIB files for SNMP and can include ldif files, make-ldif templates, schema files, `dsconfig` batch files, and other items for configuring or managing the server .
`revert-update`	The **`revert-update`** tool for UNIX/Linux systems.

Directories/Files/Tools	Description
`revert-update.bat`	The **`revert-update`** tool for Windows systems.
`setup`	The **`setup`** tool for UNIX/Linux systems.
`setup.bat`	The **`setup`** tool for Windows systems.
`scim-data-tmp`	Used to create temporary files containing System for Cross-domain Identity Management (SCIM) request data.
`uninstall`	The **`uninstall`** tool for UNIX/Linux systems.
`uninstall.bat`	The **`uninstall`** tool for Windows systems.
`update`	The **`update`** tool for UNIX/Linux systems.
`update.bat`	The **`update`** tool for Windows systems.
`Velocity`	Stores any customized Velocity templates and other artifacts (CSS, Javascript, images), or Velocity applications hosted by the server.

make-ldif template format

The `make-ldif` template demonstrates the use of all supported tags.

`make-ldif` template

> ⓘ **Note:**
>
> Because this file contains elements that are not defined in the schema, do not import it into a directory.

```
# Copyright 2012-2020 Ping Identity Corporation
# All Rights Reserved.
#
# This file provides a make-ldif template whose primary purpose is to
# demonstrate the use of all variations of all supported tags.  It is not
# intended to result in an LDIF file that should be imported into a
 directory,
# since it includes elements that aren't defined in the schema.
 Nevertheless,
# this file may be useful to individuals looking to create their own
# make-ldif templates or to customize an existing template.


# Any global replacement tokens defined at the top of the template file will
# be replaced with their corresponding value later in the template anywhere
# the token name appears in square brackets.
define basedn=dc=example,dc=com


# A branch is a special kind of entry with a fixed, non-dynamically
 generated
# DN (although the DN may be based partially or entirely on a global
# replacement token).  Some kinds of tags may be used in branch definitions
 (as
# indicated below thorugh the "allowed-in-branch" attribute.  Also, the
# "subordinateTemplate" element specifies the type and number of entries to
# create immediately below each branch.
```

```
branch: [basedn]
subordinateTemplate: attribute-value-tag-with-name:1
subordinateTemplate: attribute-value-tag-with-name-and-max-length:1
subordinateTemplate: dn-tag-full:1
subordinateTemplate: dn-tag-partial:1
subordinateTemplate: escaped-characters:1
subordinateTemplate: file-tag-path:1
subordinateTemplate: file-tag-path-sequential:1
subordinateTemplate: first-tag:1
subordinateTemplate: guid-tag:1
subordinateTemplate: ifabsent-tag-with-name:1
subordinateTemplate: ifabsent-tag-with-name-and-value:1
subordinateTemplate: ifpresent-tag-with-name:1
subordinateTemplate: ifpresent-tag-with-name-and-value:1
subordinateTemplate: last-tag:1
subordinateTemplate: list-tag:1
subordinateTemplate: list-tag-with-weights:1
subordinateTemplate: local-value-line:1
subordinateTemplate: multiple-tag-with-fixed-number-of-values:1
subordinateTemplate: multiple-tag-with-range-of-values:1
subordinateTemplate: multiple-tag-with-number-of-values-determined-from-
attribute:1
subordinateTemplate: parentdn-tag:1
subordinateTemplate: presence-tag:1
subordinateTemplate: rdn-tag:1
subordinateTemplate: random-tag-alpha-fixed-length:1
subordinateTemplate: random-tag-alpha-variable-length:1
subordinateTemplate: random-tag-numeric-fixed-length:1
subordinateTemplate: random-tag-numeric-range:1
subordinateTemplate: random-tag-numeric-range-custom-format:1
subordinateTemplate: random-tag-alphanumeric-fixed-length:1
subordinateTemplate: random-tag-alphanumeric-variable-length:1
subordinateTemplate: random-tag-chars-fixed-length:1
subordinateTemplate: random-tag-chars-variable-length:1
subordinateTemplate: random-tag-hex-fixed-length:1
subordinateTemplate: random-tag-hex-variable-length:1
subordinateTemplate: random-tag-base64-fixed-length:1
subordinateTemplate: random-tag-base64-variable-length:1
subordinateTemplate: random-tag-month:1
subordinateTemplate: random-tag-month-with-max-length:1
subordinateTemplate: random-tag-telephone:1
subordinateTemplate: random-tag-timestamp:1
subordinateTemplate: random-tag-timestamp-range:1
subordinateTemplate: sequential-tag:1
subordinateTemplate: sequential-tag-with-starting-point:1
subordinateTemplate: sequential-tag-with-starting-point-and-reset-below-new-
parent:1
subordinateTemplate: underscore-dn-tag-full:1
subordinateTemplate: underscore-dn-tag-partial:1
subordinateTemplate: underscore-parentdn-tag:1


# The template definitions come after the branch definitions, and define
# entries that will be dynamically generated, including their DNs.  Entries
# generated from a template may be either created directly below a branch,
 or
# they may be created below other template-generated entries (i.e., a
 template
# may include a "subordinateTemplate" element indicating that one or more
# entries of a different type should appear below each of the initial
 template
# entries.  In this template file, each template below will demonstrate the
# use of exactly one tag, and exactly one variation of that tag (so tags
 with
```

```
# multiple variations may be illustrated through multiple templates).

template: attribute-value-tag-with-name
rdnAttr: cn
objectClass: top
objectClass: make-ldif-example
cn: attribute value tag with name
syntax: open-curly-brace attribute-name close-curly-brace
example: {cn}
allowed-in-branch: true
description: This tag will be replaced with the first value of the indicated
 attribute.  The specified attribute must have already been assigned at
 least one value earlier in the template.

template: attribute-value-tag-with-name-and-max-length
rdnAttr: cn
objectClass: top
objectClass: make-ldif-example
cn: attribute value tag with name and max length
syntax: open-curly-brace attribute-name colon max-length close-curly-brace
example: {cn:5}
allowed-in-branch: true
description: This tag will be replaced with at most the specified number
 of characters from the beginning of the first value of the indicated
 attribute.  The specified attribute must have already been assigned at
 least one value earlier in the template.

template: dn-tag-full
rdnAttr: cn
objectClass: top
objectClass: make-ldif-example
cn: dn tag with full dn
syntax: less-than "dn" greater-than
example: <dn>
allowed-in-branch: true
description: This tag will be replaced with the full DN of the entry
 currently being generated.  All RDN attributes must be assigned values
 earlier in the template.

template: dn-tag-partial
rdnAttr: cn
objectClass: top
objectClass: make-ldif-example
cn: dn tag with partial dn
syntax: less-than "dn" colon num-components greater-than
example: <dn:1>
allowed-in-branch: true
description: This tag will be replaced with at most the specified number of
 components from the entry currently being generated, as counted from left
 to right (e.g., a value of 1 will cause only the RDN to be used, a value
 of 2 will cause both the current entry's RDN and the parent entry's RDN to
 be used, etc.).  All RDN attributes must be assigned values earlier in the
 template.

template: escaped-characters
rdnAttr: cn
objectClass: top
objectClass: make-ldif-example
cn: An example of escaping special characters
syntax: backslash-character character-to-escape
example: \{ "this" : "is JSON" \}
allowed-in-branch: true
description: The following characters need to be escaped with a backslash in
 order to be taken literally by make-ldif: \{ \} \< \> \[ \]
```

```
template: file-tag-path
rdnAttr: cn
objectClass: top
objectClass: make-ldif-example
cn: file tag with path
syntax: less-than "file" colon file-path greater-than
example: <file:cities>
allowed-in-branch: true
description: This tag will be replaced with a randomly-selected line from
 the specified file.  The file may be specified using a full path, but if no
 path information is provided, then it will be assumed to be in the config/
MakeLDIF directory below the server root.

template: file-tag-path-sequential
rdnAttr: cn
objectClass: top
objectClass: make-ldif-example
cn: file tag with path and sequential
syntax: less-than "file" colon file-path colon "sequential" greater-than
example: <file:cities:sequential>
allowed-in-branch: true
description: Each subsequent use of this tag will be replaced with the next
 line from the specified file.  Once all lines of the file have been used,
 then subsequent uses will start over from the beginning of the file.

template: first-tag
rdnAttr: cn
objectClass: top
objectClass: make-ldif-example
cn: first tag
syntax: less-than "first" greater-than
example: <first>
allowed-in-branch: false
description: This tag will be replaced with a first name value (taken
 from the first.names file).  The algorithm that the make-ldif tool uses
 guarantees that the combination of the first and last name values will
 always be unique, no matter how many entries are generated (although it may
 need to append a number to the last name value in LDIF files with extremely
 large numbers of values).

template: guid-tag
rdnAttr: cn
objectClass: top
objectClass: make-ldif-example
cn: GUID tag
syntax: less-than "guid" greater-than
example: <guid>
allowed-in-branch: true
description: This tag will be replaced with a randomly-generated globally-
unique identifier (GUID), also known as a universally-unique identifier
 (UUID).

template: ifabsent-tag-with-name
rdnAttr: cn
objectClass: top
objectClass: make-ldif-example
cn: ifabsent tag with name
syntax: less-than "ifabsent" colon attribute-name greater-than
example: <ifabsent:cn>This value will only appear if the resulting entry
 doesn't have a cn attribute.
allowed-in-branch: true
description: This tag is not intended to generate replacement text, and will
 always be replaced with the empty string.  However, it may cause any in
```

```
 which this tag is used to be completely suppressed from the resulting entry
 if the specified attribute already exists in the generated entry.  This is
 primarily useful for attributes whose inclusion in generated entries is not
 guaranteed (e.g., randomly determined via the presence tag).

template: ifabsent-tag-with-name-and-value
rdnAttr: cn
objectClass: top
objectClass: make-ldif-example
cn: ifabsent tag with name
syntax: less-than "ifabsent" colon attribute-name colon attribute-value
 greater-than
example: <ifabsent:cn:value>This value will only appear if the resulting
 entry doesn't have a cn attribute with a value of "value".
allowed-in-branch: true
description: This tag is not intended to generate replacement text, and will
 always be replaced with the empty string.  However, it may cause any in
 which this tag is used to be completely suppressed from the resulting entry
 if the specified attribute value already exists in the generated entry.
  This is primarily useful for attributes whose inclusion in generated
 entries is not guaranteed (e.g., randomly determined via the presence tag).

template: ifpresent-tag-with-name
rdnAttr: cn
objectClass: top
objectClass: make-ldif-example
cn: ifpresent tag with name
syntax: less-than "ifpresent" colon attribute-name greater-than
example: <ifpresent:cn>This value will only appear if the resulting entry
 includes a cn attribute.
allowed-in-branch: true
description: This tag is not intended to generate replacement text, and will
 always be replaced with the empty string.  However, it may cause any in
 which this tag is used to be completely suppressed from the resulting entry
 if the specified attribute is missing from the generated entry.  This is
 primarily useful for attributes whose inclusion in generated entries is not
 guaranteed (e.g., randomly determined via the presence tag).

template: ifpresent-tag-with-name-and-value
rdnAttr: cn
objectClass: top
objectClass: make-ldif-example
cn: ifpresent tag with name
syntax: less-than "ifpresent" colon attribute-name colon attribute-value
 greater-than
example: <ifpresent:cn:value>This value will only appear if the resulting
 entry includes a cn attribute with a value of "value".
allowed-in-branch: true
description: This tag is not intended to generate replacement text, and will
 always be replaced with the empty string.  However, it may cause any in
 which this tag is used to be completely suppressed from the resulting entry
 if the specified attribute value is missing from the generated entry.  This
 is primarily useful for attributes whose inclusion in generated entries is
 not guaranteed (e.g., randomly determined via the presence tag).

template: last-tag
rdnAttr: cn
objectClass: top
objectClass: make-ldif-example
cn: last tag
syntax: less-than "last" greater-than
example: <last>
allowed-in-branch: false
```

```
description: This tag will be replaced with a last name value (taken
 from the last.names file).  The algorithm that the make-ldif tool uses
 guarantees that the combination of the first and last name values will
 always be unique, no matter how many entries are generated (although it may
 need to append a number to the last name value in LDIF files with extremely
 large numbers of values).

template: list-tag
rdnAttr: cn
objectClass: top
objectClass: make-ldif-example
cn: list tag
syntax: less-than "list" colon value1 colon value2 ... colon valueN greater-
than
example: <list:first:second:third>
allowed-in-branch: true
description: This tag will be replaced with one of the values in the given
 list.  The value will be selected at random, with each possibility having
 the same chance of being selected.

template: list-tag-with-weights
rdnAttr: cn
objectClass: top
objectClass: make-ldif-example
cn: list tag with weights
syntax: less-than "list" colon value1 semicolon weight1 colon value2
 semicolon weight2 ... colon valueN semicolon weightN greater-than
example: <list:first;50:second;30:third;20>
allowed-in-branch: true
description: This tag will be replaced with one of the values in the given
 list.  The value will be selected at random, with the relative weight for
 each value used to determine its likelihood of being selected.  The weight
 values do not need to add up to 100, but if they do add up to 100 then the
 weights will be treated as percentages.

template: local-value-line
rdnAttr: cn
objectClass: top
objectClass: make-ldif-example
cn: local value definition line
syntax: local name=tags
local streetAddress=<random:numeric:5> <file:streets> Street
local locality=<file:cities>
local region=<file:states>
local postalCode=<random:numeric:5>
examplePostalAddressJSON: \{"formatted":"{streetAddress}, {locality},
 {region}, {postalCode},
 US","streetAddress":"{streetAddress}","locality":"{locality}","region":"{region}","post
allowed-in-branch: true
description: This line will define a local value that can be accessed using
 \{name\} just like attribute accesses. The local is evaluated strictly on
 each template entry creation.

template: multiple-tag-with-fixed-number-of-values
rdnAttr: cn
objectClass: top
objectClass: make-ldif-example
cn: multiple tag with fixed number of values
syntax: less-than "multiple" colon num-values greater-than
example: <multiple:5><random:alpha:10>
allowed-in-branch: true
description: This tag will cause make-ldif to attempt to generate the
 specified number of different values for this attribute.  Any line
```

```
 including this tag must have a value that includes some variable content so
 that all of the values will be different.

template: multiple-tag-with-range-of-values
rdnAttr: cn
objectClass: top
objectClass: make-ldif-example
cn: multiple tag with range of values
syntax: less-than "multiple" colon min-values colon max-values greater-than
example: <multiple:1:5><random:alpha:10>
allowed-in-branch: true
description: This tag will cause make-ldif to attempt to generate a number
 of different values for this attribute, with the number of values to
 generate selected at random between the specified minimum and maximum
 bounds.  Any line including this tag must have a value that includes some
 variable content so that all of the values will be different.

template: multiple-tag-with-number-of-values-determined-from-attribute
rdnAttr: cn
objectClass: top
objectClass: make-ldif-example
cn: multiple tag with fixed number of values
syntax: less-than "multiple" colon open-curly-brace attribute-name close-
curly-brace greater-than
random-integer: <random:numeric:1:5>
example: <multiple:{random-integer}><random:alpha:10>
allowed-in-branch: true
description: This tag will cause make-ldif to attempt to generate a number
 of different values for this attribute, with the number of values to
 generate determined from another attribute already defined in the entry
 (which may itself be dynamically-generated).  Any line including this tag
 must have a value that includes some variable content so that all of the
 values will be different.

template: parentdn-tag
rdnAttr: cn
objectClass: top
objectClass: make-ldif-example
cn: parentdn tag
syntax: less-than "parentdn" greater-than
example: <parentdn>
allowed-in-branch: false
description: This tag will be replaced with the full DN of the entry
 immediately superior to the entry being generated.

template: presence-tag
rdnAttr: cn
objectClass: top
objectClass: make-ldif-example
cn: presence tag
syntax: less-than "presence" colon percent greater-than
example: <presence:50>There is a 50% chance that this value will appear in
 the generated entry.
allowed-in-branch: true
description: This tag is not intended to generate replacement text,a nd
 will always be replaced with the empty string.  However, it may be used
 to control whether the entire attribute value will appear in the resulting
 entry.

template: rdn-tag
rdnAttr: cn
objectClass: top
objectClass: make-ldif-example
cn: rdn tag
```

```
syntax: less-than "rdn" greater-than
example: <rdn>
allowed-in-branch: true
description: This tag will be replaced with the RDN of the entry being
 generated.  All RDN attributes must be assigned values earlier in the
 template.

template: random-tag-alpha-fixed-length
rdnAttr: cn
objectClass: top
objectClass: make-ldif-example
cn: random tag alpha fixed length
syntax: less-than "random" colon "alpha" colon length greater-than
example: <random:alpha:5>
allowed-in-branch: true
description: This tag will be replaced with a string containing the
 specified number of randomly-selected lowercase alphabetic characters.

template: random-tag-alpha-variable-length
rdnAttr: cn
objectClass: top
objectClass: make-ldif-example
cn: random tag alpha variable length
syntax: less-than "random" colon "alpha" colon min-length colon max-length
 greater-than
example: <random:alpha:5:10>
allowed-in-branch: true
description: This tag will be replaced with a string containing a number
 of randomly-selected lowercase alphabetic characters.  The length of the
 string will be selected at random between the specified minimum and maximum
 length.

template: random-tag-numeric-fixed-length
rdnAttr: cn
objectClass: top
objectClass: make-ldif-example
cn: random tag numeric fixed length
syntax: less-than "random" colon "numeric" colon length greater-than
example: <random:numeric:5>
allowed-in-branch: true
description: This tag will be replaced with a string containing the
 specified number of randomly-selected numeric digits.

template: random-tag-numeric-range
rdnAttr: cn
objectClass: top
objectClass: make-ldif-example
cn: random tag numeric range
syntax: less-than "random" colon "numeric" colon lower-bound colon upper-
bound greater-than
example: <random:numeric:1:10>
allowed-in-branch: true
description: This tag will be replaced with a numeric value selected at
 random between the specified lower and upper bounds.

template: random-tag-numeric-range-custom-format
rdnAttr: cn
objectClass: top
objectClass: make-ldif-example
cn: random tag numeric range custom format
syntax: less-than "random" colon "numeric" colon lower-bound colon upper-
bound colon format greater-than
example: <random:numeric:1:1000:0000>
allowed-in-branch: true
```

```
description: This tag will be replaced with a numeric value selected at
 random between the specified lower and upper bounds.  It will use the
 specified format string (as used by the java.text.DecimalFormat class) to
 format the value.

template: random-tag-alphanumeric-fixed-length
rdnAttr: cn
objectClass: top
objectClass: make-ldif-example
cn: random tag alphanumeric fixed length
syntax: less-than "random" colon "alphanumeric" colon length greater-than
example: <random:alphanumeric:5>
allowed-in-branch: true
description: This tag will be replaced with a string containing the
 specified number of randomly-selected lowercase alphabetic characters and
 numeric digits.

template: random-tag-alphanumeric-variable-length
rdnAttr: cn
objectClass: top
objectClass: make-ldif-example
cn: random tag alphanumeric variable length
syntax: less-than "random" colon "alphanumeric" colon min-length colon max-
length greater-than
example: <random:alphanumeric:5:10>
allowed-in-branch: true
description: This tag will be replaced with a string containing a number
 of randomly-selected lowercase alphabetic characters and numeric digits.
 The length of the string will be selected at random between the specified
 minimum and maximum length.

template: random-tag-chars-fixed-length
rdnAttr: cn
objectClass: top
objectClass: make-ldif-example
cn: random tag chars fixed length
syntax: less-than "random" colon "chars" colon characters colon length
 greater-than
example: <random:chars:abcdef:10>
allowed-in-branch: true
description: This tag will be replaced with a string containing the
 specified number of characters selected at random from the provided
 character set.

template: random-tag-chars-variable-length
rdnAttr: cn
objectClass: top
objectClass: make-ldif-example
cn: random tag chars variable length
syntax: less-than "random" colon "chars" colon characters colon min-length
 colon max-length greater-than
example: <random:chars:abcdef:5:10>
allowed-in-branch: true
description: This tag will be replaced with a string containing a number of
 characters selected at random from the provided character set.  The length
 of the string will be selected at random between the specified minimum and
 maximum length.

template: random-tag-hex-fixed-length
rdnAttr: cn
objectClass: top
objectClass: make-ldif-example
cn: random tag hex fixed length
syntax: less-than "random" colon "hex" colon length greater-than
```

```
example: <random:hex:10>
allowed-in-branch: true
description: This tag will be replaced with a string containing the
 specified number of randomly-selected hexadecimal digits.

template: random-tag-hex-variable-length
rdnAttr: cn
objectClass: top
objectClass: make-ldif-example
cn: random tag hex variable length
syntax: less-than "random" colon "hex" colon min-length colon max-length
 greater-than
example: <random:hex:5:10>
allowed-in-branch: true
description: This tag will be replaced with a string containing a number
 of randomly-selected hexadecimal digits.  The length of the string will be
 selected at random between the specified minimum and maximum length.

template: random-tag-base64-fixed-length
rdnAttr: cn
objectClass: top
objectClass: make-ldif-example
cn: random tag base64 fixed length
syntax: less-than "random" colon "base64" colon length greater-than
example: <random:base64:10>
allowed-in-branch: true
description: This tag will be replaced with a string containing the
 specified number of randomly-selected characters from the base64 character
 set.

template: random-tag-base64-variable-length
rdnAttr: cn
objectClass: top
objectClass: make-ldif-example
cn: random tag base64 variable length
syntax: less-than "random" colon "base64" colon min-length colon max-length
 greater-than
example: <random:base64:5:10>
allowed-in-branch: true
description: This tag will be replaced with a string containing a number of
 randomly-selected characters from the base64 character set.  The length of
 the string will be selected at random between the specified minimum and
 maximum length.

template: random-tag-month
rdnAttr: cn
objectClass: top
objectClass: make-ldif-example
cn: random tag month
syntax: less-than "random" colon "month" greater-than
example: <random:month>
allowed-in-branch: true
description: This tag will be replaced with a string containing the name of
 a randomly-selected month of the year.  The American English name of the
 month will be used, with the first letter capitalized.

template: random-tag-month-with-max-length
rdnAttr: cn
objectClass: top
objectClass: make-ldif-example
cn: random tag month with max length
syntax: less-than "random" colon "month" colon max-length greater-than
example: <random:month:3>
allowed-in-branch: true
```

```
description: This tag will be replaced with a string containing at most
 the specified number of characters from the beginning of the name of a
 randomly-selected month of the year.  The American English name of the
 month will be used, with the first letter capitalized.

template: random-tag-telephone
rdnAttr: cn
objectClass: top
objectClass: make-ldif-example
cn: random tag telephone
syntax: less-than "random" colon "telephone" greater-than
example: <random:telephone>
allowed-in-branch: true
description: This tag will be replaced with a string containing a randomly-
generated telephone number, using the international representation of a
 U.S. telephone number in the format "+1 xxx xxx xxxx" (in which each x
 represents a randomly-selected numeric digit).

template: random-tag-timestamp
rdnAttr: cn
objectClass: top
objectClass: make-ldif-example
cn: random tag timestamp
syntax: less-than "random" colon "timestamp" greater-than
example: <random:timestamp>
allowed-in-branch: true
description: This tag will be replaced with a string containing a
 generalized time representation of a randomly-selected time within the last
 ten years.

template: random-tag-timestamp-range
rdnAttr: cn
objectClass: top
objectClass: make-ldif-example
cn: random tag timestamp range
syntax: less-than "random" colon "timestamp" colon min-timestamp colon max-
timestamp greater-than
example: <random:timestamp:20000101000000.000Z:20091231235959.999Z>
allowed-in-branch: true
description: This tag will be replaced with a string containing a
 generalized time representation of a randomly-selected time within the
 specified time range (in which the minimum and maximum values should also
 be specified using the generalized time format).

template: sequential-tag
rdnAttr: cn
objectClass: top
objectClass: make-ldif-example
cn: sequential tag
syntax: less-than "sequential" greater-than
example: <sequential>
allowed-in-branch: true
description: This tag will be replaced with a sequentially-incrementing
 value such that each subsequent entry will have a value that is greater
 than the one in the entry before it.  The first value generated will be
 zero.

template: sequential-tag-with-starting-point
rdnAttr: cn
objectClass: top
objectClass: make-ldif-example
cn: sequential tag with starting point
syntax: less-than "sequential" colon initial-value greater-than
example: <sequential:1>
```

```
allowed-in-branch: true
description: This tag will be replaced with a sequentially-incrementing
 value such that each subsequent entry will have a value that is greater
 than the one in the entry before it.  The first value generated will be the
 specified initial value.

template: sequential-tag-with-starting-point-and-reset-below-new-parent
rdnAttr: cn
objectClass: top
objectClass: make-ldif-example
cn: sequential tag with starting point and reset below new parent
syntax: less-than "sequential" colon initial-value colon "true" greater-than
example: <sequential:1:true>
allowed-in-branch: true
description: This tag will be replaced with a sequentially-incrementing
 value such that each subsequent entry will have a value that is greater
 than the one in the entry before it.  The first value generated will be the
 specified initial value.  If this template is used below multiple parents,
 then the counter will be reset to the initial value for the first entry
 below each new parent.

template: underscore-dn-tag-full
rdnAttr: cn
objectClass: top
objectClass: make-ldif-example
cn: underscore dn tag with full dn
syntax: less-than "_dn" greater-than
example: <_dn>
allowed-in-branch: true
description: This tag will be replaced with the full DN of the entry
 currently being generated, except with commas replaced with underscores.
 All RDN attributes must be assigned values earlier in the template.

template: underscore-dn-tag-partial
rdnAttr: cn
objectClass: top
objectClass: make-ldif-example
cn: underscore dn tag with partial dn
syntax: less-than "_dn" colon num-components greater-than
example: <_dn:1>
allowed-in-branch: true
description: This tag will be replaced with at most the specified number of
 componenets from the entry currently being generated, as counted from left
 to right (e.g., a value of 1 will cause only the RDN to be used, a value
 of 2 will cause both the current entry's RDN and the parent entry's RDN
 to be used, etc.), except with commas replaced with underscores.  All RDN
 attributes must be assigned values earlier in the template.

template: underscore-parentdn-tag
rdnAttr: cn
objectClass: top
objectClass: make-ldif-example
cn: underscore-parentdn tag
syntax: less-than "_parentdn" greater-than
example: <_parentdn>
allowed-in-branch: false
description: This tag will be replaced with the full DN of the entry
 immediately superior to the entry being generated, except with commas
 replaced with underscores.
```

Server installation modes

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

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

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

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

Before you begin

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

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

> ⓘ **Important:**

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

Ping license keys

License keys are required to install, update, and renew all Ping products.

How to obtain a license

To obtain a license key, contact your account representative or use the *Ping Identity licensing portal*.

When do you need a license

A license is required for setting up a new single server instance and can be used site-wide for all servers in an environment. Additionally, you must obtain a new license when updating a server to a new major version, such as when upgrading from 7.3 to 8.0. When cloning a server instance with a valid license, you do not need a new license.

ⓘ **Note:**

The update process displays a prompt for a new license.

How to specify a license

- Specify a license at setup

  You have these options:

  - Use the `--licenseKeyFile` <*path-to-license*> option with **`setup`**.
  - Copy the license file to the server root directory and then run the **`setup`** tool. The tool discovers the license file.
- Specify a license after setup

  Use the Administrative Console or **`dsconfig`** (in the Topology section, select License).

  ⓘ **Note:** Placing the new license file in the server root directory does not work in this case.

How to view the license status

To view the details of a license, including its expiration, you have these options:

- The server's **`status`** tool
- The Administrative Console's **Status** page (On the **Monitors** tab, search for License.)

License expiration

The server provides a notification as the expiration date approaches.

Before a license expires, obtain a new one and install it by using **`dsconfig`** or the Administrative Console.

ⓘ **Note:**

An expiring license causes alerts and alarms but does not affect the functionality of the product.

Setting up the Directory Server in interactive mode

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

About this task

Complete the following steps to install the Directory Server in interactive mode.

Steps

1. Extract the distribution `.zip` file, review *Before You Begin*, and then go to the server root directory.

   ```
   $ ./setup
   ```

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

   ```
   $ env JAVA_HOME=/ds/java ./setup
   ```

2. Read the Ping Identity End-User License Agreement, and type `yes` to continue.
3. Enter the fully qualified host name or IP address of the local host, or press Enter to accept the default.
4. Enter the distinguished name (DN) for the initial root user, or press Enter to accept the default (`cn=Directory Manager`).
5. Enter and confirm the root user password.
6. Press Enter to enable the Ping Identity services (Configuration, Consent, Delegated Admin, Documentation, and Directory REST API) and Administrative Console over HTTPS.

   After setup, you can enable or disable individual services and applications by configuring the HTTPS Connection Handler.
7. Enter the port on which the Directory Server will accept connections from HTTPS clients, or press Enter to accept the default.
8. Enter the port on which the Directory Server will accept connections from LDAP clients, or press Enter to accept the default.
9. Enter `no` to use a standard LDAP connection, or accept the default (`yes`) to enable both LDAPS and StartTLS.

   Enabling LDAPS configures the LDAPS Connection Handler to allow SSL over its client connections. Enabling StartTLS configures the LDAP Connection Handler to allow StartTLS.
10. Select the certificate option for this server:
    - Generate a self-signed certificate for testing purposes only.
    - Enter the keystore path and keystore PIN to use an existing certificate using a Java Keystore, .
    - Enter the keystore path and the keystore PIN to use an existing certificate using use a PKCS#12 keystore.
    - Enter only the keystore PIN ro use the PKCS#11 token.

11. Choose the desired encryption for the directory data, backups, and log files from the choices provided:
    - Encrypt data with a key generated from an interactively provided passphrase. Using a passphrase (obtained interactively or read from a file) is the recommended approach for new deployments. Use the same encryption passphrase when setting up each server in the topology.
    - Encrypt data with a key generated from a passphrase read from a file.
    - Encrypt data with a randomly generated key. This option is primarily intended for testing purposes, especially when only testing with a single instance, or if you intend to import the resulting encryption settings definition into other instances in the topology.
    - Encrypt data with an imported encryption settings definition. This option is recommended if you are adding a new instance to an existing topology that has older server instances with data encryption enabled.
    - Do not encrypt server data.
12. Type the base DN for the data, or accept the default base DN of `dc=example,dc=com`.
13. To choose an option to generate and import sample data, type the desired number of entries, or press Enter to accept the default number (10000).

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

    Priming the cache can increase the startup time for the Directory Server but provides optimum performance after startup has completed. This option is best used for strict throughput or response time performance requirements, or if other replicas in a replication topology can accept traffic while this Directory Server instance is starting. Priming the cache also helps determine the recommended JVM option, CMSInitiatingOccupancyFraction, when a Java garbage collection pause occurs. See *JVM Garbage Collection Using CMS*.
16. Enter a location name for this server.
17. Enter a unique instance name for this server.

    You cannot change the name after you set it.
18. Press Enter to accept the default (yes) to start the Directory Server after the configuration has completed.

    To configure additional settings or import data, enter `no` to keep the server in shutdown mode.
19. Choose an option to continue server set up.
20. In the **Setup Summary** window, confirm the configuration. , or
    - Press Enter to accept the default (set up with the parameters given), enter the option to repeat the installation process.
    - Enter the option to cancel the setup completely.

Installing the Directory Server in non-interactive mode

Run the **`setup`** command in non-interactive mode to automate the installation process using a script or to run the command directly from the command line.

Non-interactive mode is useful when setting up production or QA servers with specific configuration requirements.

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

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

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

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

Installing the Directory Server in non-interactive mode
Install a Directory Server in a production or QA environment with no security enabled.

Steps

- Extract the distribution `.zip` file, review “Before You Begin”, and then use **`setup`** with the `--cli` and `--no-prompt` options for non-interactive mode from the `<server-root>` directory.

  The following command uses the default root user distinguished name (DN) (cn=Directory Manager) with the specified `--rootUserPassword` option. You must include the `--acceptLicense` option or the setup generates an error message. The --instancename option specifies the name for the server instance and should be unique across all instances in the topology. The --location option specifies the name of the location in which the instance will be installed. You should generally configure your topology with a separate location for each data center to allow inter-server communication to prioritize servers in the same location over those in remote locations.

  ```
  $ ./setup --cli --no-prompt --rootUserPassword "password" \
    --baseDN "dc=example,dc=com" --acceptLicense --ldapPort 389 \
    --instancename Instance1 --location Location1
  ```

Installing the Directory Server in non-interactive mode with a truststore

About this task

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

Steps

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

  ```
  $ ./setup --cli --no-prompt --rootUserPassword "password" \
    --baseDN "dc=example,dc=com" --ldapPort 389 --enableStartTLS \
    --ldapsPort 636 --useJavaKeystore config/keystore.jks \
    --keyStorePasswordFile config/keystore.pin \
    --certNickName server-cert --useJavaTrustStore config/truststore.jks \
    --acceptLicense --instancename Instance1 --location Location1
  ```

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

Installing a lightweight server

Users who want to demo or test a lightweight version of the Directory Server on a memory-restricted machine can do so by removing all unused or unneeded configuration objects.

All configuration entries, whether enabled or not, take up some amount of memory to hold the definition and listeners that are notified of changes to those objects.

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

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

Reduce the number of worker threads

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

```
$ bin/dsconfig set-work-queue-prop \
  --set num-worker-threads:8 \
  --set num-administrative-session-worker-threads:4 \
  --set max-work-queue-capacity:100
```

Reduce the percentage of JVM memory used for the JE database cache

When you have a memory-constrained environment, you want to ensure that as much of the memory that is there is available for use during processing and not tied up caching database contents.

```
$ bin/dsconfig set-backend-prop --backend-name userRoot --set db-cache-
percent:5
```

Disable the Dictionary Password Validator

The Dictionary Password Validator takes a lot of memory to hold its dictionary. Disabling it frees up some memory. You can delete the other password validators if not needed, such as Attribute Value, Character Set, Length-based, Repeated Characters, Similarity-based, or Unique Characters Password Validator.

```
$ bin/dsconfig delete-password-validator --validator-name Dictionary
```

Remove non-essential schema files

Although not recommended for production deployments, some candidates that you can remove are the following: `03-rfc2713.ldif`, `03-rfc2714.ldif`, `03-rfc2739.ldif`, `03-rfc2926.ldif`, `03-rfc2985.ldif`, `03-rfc3712.ldif`, `03-uddiv3.ldif`.

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

Uninstalling the Server

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

To uninstall a server instance, run the **setup** tool in interactive command-line, or non-interactive command-line mode.

Interactive command-line mode

Interactive command-line mode is a text-based interface that prompts the user for input. To run this mode, use the **bin/uninstall** command with the `--cli` option.

Non-interactive command-line mode

Non-interactive command-line mode suppresses progress information from being written to standard output during processing, except for fatal errors. To run this mode, use the **bin/uninstall** command with the `--no-prompt` option.

ⓘ **Note:**

For standalone installations with a single Directory Server instance, you can also manually remove the Directory Server by stopping the server and recursively deleting the directory and subdirectories, as in the following example.

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

Uninstalling the server in interactive mode

Interactive mode uses a text-based, command-line interface to help you remove your instance.

About this task

If **uninstall** cannot remove all of the Directory Server files, the **uninstall** tool generates a message with a list of the files and directories that must be manually deleted. The **uninstall** command must be run as either the root user or the same user (or role) that installed the Directory Server.

Steps

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

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

2. Select the components to be removed.
   - Remove all components - If you want to remove all components, press **Enter** to accept the default (remove all).
   - Select the components to be removed - If you do not want to remove all components, enter the option to specify the removal of only specific components.

     For each type of server component, press **Enter** to remove it or enter `no` to keep it.
3. If the Directory Server is part of a replication topology, enter `yes` to provide your authentication credentials (Global Administrator ID and password). If you are uninstalling a standalone server, continue to step 7.
4. Enter the Global Administrator ID and password to remove the references to this server in other replicated servers, and then enter or verify the host name or IP address for the server that you are uninstalling.
5. Select how you want to trust the server certificate if you have set up SSL or StartTLS. Press **Enter** to accept the default.

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

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

```
Enter choice [3]:
```

6. View the logs for any remaining files and manually remove any remaining files or directories.

   If your Directory Server is running, the server shuts down before continuing the uninstall process. The uninstall action processes the removal requests before completing.

Uninstalling the server in non-interactive mode

The **uninstall** utility provides a non-interactive method to enter the command with the `--no-prompt` option.

About this task
Another useful argument is the `--forceOnError` option that continues the uninstall process when an error is encountered. If an option is incorrectly entered or if a required option is omitted and the `--forceOnError` option is not used, the command fails and aborts.

Steps

1. From the server root directory, run **uninstall** tool with the `--remove-all` option to remove all of the Directory Server's libraries.

   The optional `--quiet` option suppresses output information.

   The following command assumes that the Directory Server is standalone and not part of a replication topology.

   ```
   $ ./uninstall --cli --remove-all --no-prompt --quiet --forceOnError
   ```

2. If any files or directories remain, manually remove them.

Uninstalling selected components in non-interactive mode

Steps

1. From the server root directory, run **uninstall** with the `--backup-files` option to remove the Directory Server's backup files.

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

2. To view the other options available to remove specific components, use the `--help` or `-H` option.

Upgrading the Server

This section presents multiple update scenarios and their implications to consider when upgrading your server code.

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

Upgrade overview and considerations

The following content provides a brief description of the upgrade process and considerations that might affect your upgrade decisions.

Overview

For the upgrade process, you must download and extract a new version of the Directory Server `.zip` file on the server that will be updated. In addition, you must run the **`update`** utility with the **`--serverRoot`** or **`-R`** option value from the new root server pointing to the installation that will be upgraded.

Considerations

Consider the following when planning for and upgrading replicating servers:

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

> ⓘ **Tip:**
>
> For additional considerations, see the *Planning your upgrade guide*.

Considerations when upgrading to 8.2.0.0

> ⓘ **Important:**
>
> If you plan to upgrade servers using a mixed-version environment where one version is earlier than 7.0 and some of the servers are still using the admin backend while others have been updated to the topology registry, do not attempt to make size changes to the topology. You cannot remove any existing servers (using `dsreplication disable`) or add new servers (using `dsreplication enable`) when in this transitional state of partially-updated servers. When a topology has been completely migrated to a 7.0 or later version with the topology registry, changes to the topology size are allowed, even in mixed-version environments (for example, mixed 7.3 and 8.3).

This upgrade moves to Jetty 9.4. As a result, the HTTPS Connection Handler no longer supports TLS_RSA ciphers by default. If you use any legacy HTTPS clients that still require TLS_RSA ciphers, modify the `ssl-cipher-suite` property of the HTTPS Connection Handler to include them.

Upgrading servers in a topology

An update to the current PingDirectory Server release includes the introduction of a topology registry, which stores information that was stored previously in the admin backend, such as server instances, instance and secret keys, server groups, and administrator user accounts.

Before you begin

Before you can update and migrate the admin backend, the following conditions must be satisfied:

- Run the update tool and provide LDAP authentication options to the peer servers of the server being updated. The **`update`** tool connects to the peer servers of the server being updated to obtain the necessary information to populate the topology registry.
- On every server in the topology, configure the encryption protocol requested, either plain, TLS, StartTLS, or SASL, for an LDAP connection.
- Ensure the LDAP credentials are present on every server in the topology.
- Ensure the users related to the LDAP credentials have permissions to read from the admin backend and the config backend of every server in the topology. For example, you can use a root DN user that has `inherit-default-privileges` set to true, such as the `cn=Directory Manager` user, that exists on every server.
- The instance name is set on every server and is unique across all servers in the topology. The instance name is a server's identifier in the topology. After all servers in the topology have been updated, each server is uniquely identified by its instance name. After the name is set, it cannot be changed.

> ⓘ **Tip:**
>
> If needed, use the following command to set the instance name of a server prior to the update.
>
> ```
> $ bin/dsconfig set-global-configuration-prop \
>   --set instance-name:uniqueName
> ```

> ⓘ **Important:**
>
> This information is only applicable when updating from a server version that uses the admin backend to a server version that uses the topology registry. This is only the case when updating from a server version earlier than a 7.x to a 7.x version and is not applicable to any updates to a server version of 8.x or later.

About this task

Upgrade to a server that uses a topology registry to store network configuration, administrator user accounts, and keys information, as well as migrate the information from the previous server version.

Steps

1. To make clustered configuration changes in a mixed-version cluster, choose one of the following options:
   - Update each server to the same version.
   - Temporarily split up the cluster by changing the `cluster-name` property on the server instance configuration objects.

   > ⓘ **Note:**
   >
   > Changes to clustered configurations are not allowed in mixed-version clusters. This applies to configuration in the `cn=Cluster,cn=config` subtree and only applies to servers with matching cluster names.

2. Make clustered configuration changes again to mirror these changes across the topology.
3. Run the **dsframework** command on the server being updated.

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

> ⓘ **Important:**
>
> To run this command, you must have enabled or fixed the configuration of the LDAP Connection Handlers to support the desired connection security protocol on each server, so that its admin backend has the most up-to-date information.

Before allowing the update, the **update** tool verifies that the following conditions are satisfied on every server in the topology:

- When the first server is being updated, all other servers in the topology are online.
- When updating additional servers, all topology information was obtained from one of the servers that has already been updated.
- The users related to the provided LDAP credentials have read permissions to the config and admin backends of the peer servers.
- The instance name is set on every server and is unique across all servers in the topology.

> ⓘ **Note:**
>
> If any of these conditions or those listed in the preceding Before you begin section are not satisfied, the **update** tool lists all of the errors encountered for each server and provides instructions on how to fix them.

When all of these conditions are met, the cluster-wide configuration is synchronized on all servers in the topology.

> ⓘ **Note:**
>
> Older versions have some topology configuration under the **cn=cluster,cn=config** JSON attribute and field constraints. These items do not support mirrored cluster-wide configuration data. In an update, avoid custom configuration changes on a server being overwritten with the configuration on the mirrored subtree master. If additional configuration steps are needed, see step 4.

4. To synchronize the cluster-wide configuration data across all servers in the topology, run the **config-diff** tool on each pair of servers to determine the differences, and use **dsconfig** to update each instance using the **config-diff** output.

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

Restoring a mixed topology to a clean state
Perform the following steps to restore a mixed-version 6.2 - 7.x topology with replication problems to a clean state with all servers at the same level and with replication functioning normally.

About this task

> ⓘ **Note:**
>
> This information is only applicable when updating from a server version that uses the admin backend to a server version that uses the topology registry. This is only the case when updating from a server version earlier than 7.x to a 7.x version and is not applicable to any updates to server version 8.x or later.

If you are upgrading a Directory Server version prior to 7.3 to 7.3 or later and at least one server was previously removed from the topology, you might receive an error message similar to one of the following:

```
[09/Dec/2019:18:37:56.927 -0700] instanceName="[INSTANCE_NAME]" \
    threadID=438 category=REPLICATION severity=NOTICE msgID=118947975
 msg="The \
    replication total backlog count is 163364"
```

Or:

```
[09/Dec/2019:17:15:10.108 -0700] category=REPLICATION \
   severity=SEVERE_ERROR msgID=14942295 msg="While clearing the database 997
 \
   o=mycompany, the following error happened: (JE 7.5.12) Attempted to
 remove \
   non-existent database 997 o=mycompany DatabaseNotFoundException: (JE
 7.5.12) \
   Attempted to remove non-existent database 997 o=pingone (DbTree.java:922
 \
   DbTree.java:1183 DbTree.java:1235 Environment.java:1001
 Environment.java:1221 \
   Environment.java:1204 Environment.java:1009 ReplicationDbEnv.java:1186 \
   DomainDbEnv.java:995 ReplicationDB.java:1002 DbHandler.java:1324
 DbHandler.java:1064 \
   DbHandler.java:1042 DbHandler.java:798 DbHandler.java:723
 DirectoryThread.java:352 \
   (7.3.0.4-20191127203417.000Z-83a75e2a))"
```

Steps

1. To export a safe copy of the data, run **`bin/export-ldif`** on the primary server.
2. Start the down-level primary server and stop the upgraded replica server.
3. Run **`bin/remove-defunct-server`** on the upgraded replica server and do the following when prompted:
   a. Select the replica server itself.
   b. Supply the credentials for the primary server when prompted.
   c. Select **[c]** to continue.
   d. Enter the path to the local `changelogDB` when prompted.
4. Leave the upgraded replica server stopped and stop the down-level primary server.
5. Run **`bin/dsreplication`** on the primary server and do the following:
   a. Select **2) Manage the topology (add and remove servers)**.
   b. Select **4) Cleanup Server**.
   c. Select **[c]** to continue.
   d. Enter the path to the local `changelogDB` when prompted.

6. Start the down-level primary server and the upgraded replica server.
7. Run **bin/status** on both servers and verify that status of **Replication: disabled** in **Data Sources**.
8. Upgrade the replica server to version 7.3.0.5 or later, which contains fixes for issues 00668265 and DS-40955.
9. Upgrade the primary server to the same level as the replica server.
10. To re-enable replication between the primary and replica server, run the **bin/dsreplication** tool.
11. Re-initialize the replica server data from the primary server.
12. Run **bin/status** on both servers to verify that replication is enabled in **Data Sources** and that both servers show the same number of entries.

Upgrading the Directory Server

Perform an upgrade to the Directory Server with the following steps.

Steps

1. Download and extract the new version of the Directory Server in a location outside the existing server's installation.

   For these steps, assume the existing server installation is located in `/prod/PingDirectory` and the new server version is extracted in the `/home/stage/PingDirectory` location.
2. Run the **update** tool provided with the new server package to update the existing Directory Server.

> ⓘ **Note:**
>
> The update tool might prompt for confirmation of the server configuration changes if it detects customization and display the following code.
>
> ```
> $ /home/staging/PingDirectory/update --serverRoot /prod/PingDirectory
> ```

Reverting an update

After PingDirectory Server has been updated, you can revert to the last version or one level back using the **revert-update** tool.

About this task

The **revert-update** tool accesses a log of file actions taken by the updater to put the file system back to its prior state. If you have run multiple updates, you can run the **revert-update** tool multiple times to revert to each prior update sequentially. You can only revert back one level. For example, if you have run the update twice since first installing PingDirectory Server, you can run the **revert-update** command to revert to its previous state, then run the **revert-update** command again to return to its original state. The following steps detail reverting from 7.x to a version prior to 7.0.

When starting up the server for the first time after a revert has been run, and the necessary extra steps have been completed, the server displays warnings about "offline configuration changes," but they are not critical and do not appear on subsequent start ups.

To revert to the most recent server version:

Steps

- Use **revert-update** in the server root directory to revert back to the most recent version of the server.

  ```
  $ PingDirectory-old/revert-update
  ```

Next steps

Reverting from the 7.0 version or later to a version earlier than 7.0 using the **revert-update** command might require extra steps. This is also the case when updating or reverting from a version earlier than 6.2.0.2 to a 6.2.0.2 version or later. These steps are listed when the **update** and **revert-update** tool are run as well. Depending on your installation and configuration, you might need to perform one or more of the following tasks:

- When updating or reverting from 6.2.0.2 or later to a version earlier than 6.2.0.2, indexes might need to be rebuilt. Older versions of the server use an incompatible format for Local DB Composite Indexes. To update a server with composite indexes in the previous format, delete these indexes and re-run the update. After the update is complete, recreate the indexes and rebuild the indexes using the **rebuild-index** tool. The command for recreating an index is in the `Undo` portion of the `logs/config-audit.log` file. To later revert to an older version, delete and recreate those composite indexes again after the revert process completes.
- When updating to 7.x for the first time, instance names must be set for each server in the topology if they were not previously set. This is done with the following **dsconfig** command.

  ```
  $ bin/dsconfig --bindDN "cn=Directory Manager" \
    --bindPassword secret \
    --no-prompt set-global-configuration-prop \
    --set instance-name:<name>
  ```

- Topology information such as server instances, instance and secret keys, server groups, and administrator users have moved to the topology portion of the configuration from the **admin** backend. As long as new servers are not added to the topology after this update, you can use the **revert-update** command to return to the previous version. However, if new servers are added, then the restored **admin** backend of this server does not contain information about the new servers, and the local server cannot communicate with any other servers in the topology. New servers should not be added to the topology if reverting this update is a possibility.
- If new servers were added to the topology after the update, the new servers must be temporarily removed from the topology until all servers have been reverted to the previous version.
- When a server is reverted to a version earlier than 7.x, any servers in the topology using the topology portion of the configuration rather than the **admin** backend must know that the reverted server was downgraded to the **admin** backend. To do this, run the following **dsconfig** command on one of the servers that has not been reverted.

  ```
  $ bin/dsconfig set-server-instance-prop \
    --instance-name <Reverted server instance name> \
    --set server-version:<Version to which server is reverted>
  ```

- If the topology does not have a master server when this command is run, the **revert-update** will not succeed. In this case, you must make one of the remaining updated servers in the topology the master with the following command. This enables the chosen instance to run the **revert-update** command successfully.

  ```
  $ bin/dsconfig set-global-configuration-prop \
    --set force-as-master-for-mirrored-data:true
  ```

- The 7.x server version includes database changes that are not compatible with previous server versions, such as 6.x or earlier. To later revert to an older version, you must export the data to LDIF before performing the reversion. Then, re-import the data after the revert process completes. In addition, you must delete the `changelogDb/` and `db/changelog/` directories in the reverted server root after the revert process completes.

Getting Started with Directory Server

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

> ⓘ **Note:**
>
> The URL is based on the host name and HTTPS port specified during installation, such as https://hostname.com:443/console.

Apply Server Configurations

Apply your server configuration changes individually or using a **dsconfig** batch file. The batch file defines the Directory Server configuration tool, **dsconfig**, commands necessary to configure your server instance. For more information on using batch files, see *Using dsconfig in Batch Mode*.

If you are migrating from a Sun Java System 5.x, 6.x, 7.x directory server, you can use the **bin/migrate-sun-ds-config** command to migrate your configuration settings to this new server instance.

Import Data

Import user data using the **import-ldif** tool. The import serves as an initial test of the schema settings.

```
$ bin/import-ldif --backendID userRoot --ldifFile ../user-data.ldif
```

Install and Configure the Delegated Admin Application

Install a Javascript-based web application for business users to manage identities stored in the Directory Server. The application provides delegated administration of identities for help desk or customer service representatives initiating a password reset and unlock, an employee in HR updating an address stored within another employee profile, or an application administrator updating identity attributes or group membership to allow application single sign-on (SSO) access.

Run Performance Tests

The Directory Server provides two tools for functional performance testing using in-house LDAP clients that access the server directly: **searchrate** to test search performance and **modrate** to test modification performance.

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

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

Multiple backends

You can create multiple local database backends, each containing one or more different base distinguished names (DNs).

There should be at most one replicating domain on each local database backend. The replication domain should not span multiple local database backends. The typical entry-balancing configuration involves two local database backends: one to serve the global domain data that resides above the entry-

balancing point and another that is defined with the entry-balancing point as the base DN, such as `ou=people,dc=example,dc=com`.

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

When creating multiple database backends, consider the following:

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

Importing data

After installing the database, such as userRoot, import data into the database.

Steps

- To add a server to a replicating set:
  a. Perform a **`dsreplication enable`** operation.
  b. Import the database through the **`dsreplication initialize`** operation.
- To add a server to a non-replicating set or to add the first server of a future replicating set, import the data with the **`bin/import-ldif`** tool.

  For more information about the **`bin/import-ldif`** tool, see *Importing and Exporting Data* on page 460.

Generating sample data

PingDirectory Server provides LDIF templates to generate sample entries for initializing your server. You can generate the sample data with the **`make-ldif`** utility together with template files that come bundled with the `.zip` build, or you can use template files that you create yourself.

About this task

The templates create sequential entries for testing PingDirectory Server with a range of dataset sizes. The Directory Server templates are located in the `config/MakeLDIF`.

The sample data templates generate a dataset with basic access control privileges that grants anonymous read access to anyone, grants users the ability to modify their own accounts, and grants the admin account full privileges. The templates also include the `uid=admin and ou=People` entries.

Steps

- To generate randomized sample data, use the `--randomSeed` option with the **`make-ldif`** command.

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

> ⓘ **Note:**
>
> If the `--randomSeed` option is used with the same seed value, the template always generates the same `.ldif` file.

  The command generates 10,000 sample entries and writes them to an output file, `data.ldif`. The random seed generator is set to 0.
- To bypass the **`make-ldif`** command, use the `--templateFile` option with the **`import-ldif`** tool.

Importing data on the Directory Server using offline import

Steps

1. Create a `.ldif` file that contains entries or locate an existing file.

   The **`import-ldif`** tool requires a `.ldif` file, which conforms to standard LDIF syntax without change records. The `changeType` attribute is not allowed in the input LDIF. For information on adding entries to , see *Managing Entries*.
2. Stop the Directory Server.
3. To import data from an LDIF file to the Directory Server, use the **`import-ldif`** command.

   > ⓘ **Tip:**
   >
   > For assistance with the list of options, run **`import-ldif --help`**.

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

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

Running the server

Run the server as a background or foreground process.

Steps

1. To start the Directory Server, run the bin/start-server command on UNIX or Linux systems.

   An analogous command is in the `bat` folder on Microsoft Windows systems.

   > ⓘ **Note:**
   >
   > The bin/start-server command starts the Directory Server as a background process when no options are specified.
2. Optional: To run the Directory Server as a foreground process, use the bin/start-server command with the `--nodetach` option.

Starting the Directory Server
Start the directory server using the terminal.

Steps

- To start the server, on the terminal run bin/start-server.

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

Running the server as a foreground process
Use the terminal to run the PingDirectory service as a foreground process.

Steps

1. To launch the Directory Server as a foreground process, open the terminal and enter bin/start-server with the `--nodetach` option.

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

2. To stop the Directory Server:
   - Press **CNTRL+C** in the terminal window where the server is running.
   - Run the bin/stop-server command from another terminal window.

Starting the server at boot time
By default,PingDirectory Server does not start automatically when the system is booted. Instead, you must manually start it with the bin/start-server command.

About this task
To configure the Directory Server to start automatically when the system boots, use the **create-systemd-script** utility to create a script, or create the script manually.

Steps

1. Create the service unit configuration file in a temporary location, where "ds" is the user running PingDirectory.

```
$ bin/create-systemd-script \
     --outputFile /tmp/ping-directory.service \
     --userName ds
```

2. As a root user, copy the **ping-directory.service** configuration file into the `/etc/systemd/system` directory.
3. To read the new configuration file, reload **systemd**.

```
$ systemctl daemon-reload
```

4. To start the PingDirectory, run the **start** command.

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

5. To configure the PingDirectory to start automatically when the system boots, run the **enable** command.

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

6. Log out as root.

   To perform this task on an RC system, create the startup script with **bin/create-rc-script** and move it to the /etc/init.d directory.

> ⓘ **Note:**
>
> Create symlinks to this script from the /etc/rc3.d directory (starting with an “S” to ensure that the server is started) and /etc/rc0.d directory (starting with a “K” to ensure that the server is stopped).

Signing on to the Administrative Console

After the server is installed, access the administrative console, **https://hostname:HTTPport/console/login**, to verify the configuration and manage the server.

To sign on to the administrative console, use the initial root user DN specified during setup (by default **cn=Directory Manager**).

You can use the **dsconfig** command or the administrative console to create additional root DN users in **cn=Root DNs,cn=config**. These new users require the fully qualified DN as the username, such as **cn=new-admin,cn=Root DNs,cn=config**. To use a simple username (without the **cn=** prefix) for signing on to the administrative console, the root DN user must have the **alternate-bind-dn** attribute configured with an alternate name, such as "admin."

The default link to the administrative console is **https://hostname:HTTPport/console/login**.

If you need to run the administrative console in an external container, such as Tomcat, you can install a separate package (/server-root/resource/admin-console.zip) according to that container's documentation.

> ⓘ **Note:**
>
> The default session timeout for the console is 24 hours. When this duration is exceeded, all inactive users are signed off automatically.
>
> To set a different timeout value, configure the server.sessionTimeout application parameter, which specifies the timeout duration in seconds. You can set the value as an init parameter either in the console or on the command line, as shown below.
>
> For changes to take effect, restart the HTTP(S) Connection Handler or the server.

Console

1. Go to **Web Application Extensions**# **Console**.
2. Use the **Init Parameter** field.

Command line

- The following example uses a value of 1800 seconds (30 minutes).

```
dsconfig set-web-application-extension-prop --no-prompt \
--extension-name Console \
--add init-parameter:server.sessionTimeout=1800
```

Stopping the Directory Server

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

About this task

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

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

Steps

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

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

Scheduling a server shutdown

Schedule a server shutdown.

Steps

- To schedule a server shutdown, use the **`bin/stop-server`** tool with the `--stopTime YYYYMMDDhhmmss` option .

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

```
$ bin/stop-server --stopTime 20120606134500Z --port 1389 \
  --bindDN "uid=admin,dc=example,dc=com" --bindPassword secret \
  --stopReason "Scheduled offline maintenance"
```

Restarting the server

Restart the Directory Server.

About this task

Running the command is equivalent to shutting down the server, exiting the Java Virtual Machine (JVM) session, and then starting up again.

Steps

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

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

Running the server as a Microsoft Windows service

The server can run as a Windows service on Windows Server 2012 R2 and Windows Server 2016. This enables signing-out of a machine without stopping the server.

Registering the server as a Windows service

Register the server as a Windows service using the Windows command prompt.

About this task

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

Steps

1. Stop the server with `bin/stop-server`.

   You cannot register a server while it is running.
2. To register the server as a service, from a Windows command prompt, run `bat/register-windows-service.bat`.
3. After registration, start the server from the Windows Services Control Panel or with the `bat/start-server.bat` command.

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

Running multiple service instances

Only one instance of a particular service can run at one time.

About this task

Services are distinguished by the `wrapper.name` property in the `<server-root>/config/wrapper-product.conf` file.

Steps

1. To run additional service instances, change the `wrapper.name` property on each additional instance.
2. Add or change descriptions of the service in the `wrapper-product.conf` file.

Deregistering and uninstalling services

To uninstall a service, you must first deregister it.

About this task

While a server is registered as a service, it cannot run as a non-service process or be uninstalled.

Steps

1. To remove the service from the Windows registry, use the `bat/deregister-windows-service.bat` file.
2. To uninstall the server, run the `uninstall.bat` script.

Configuring log files for services

About this task

The log files are stored in *`<server-root>`*`/logs`, and file names begin with `windows-service-wrapper`. They are configured to rotate each time the wrapper starts or the file reaches its maximum size. Only the last three log files are retained.

Steps

These configurations can be changed in the *`<server-root>`*`/config/wrapper.conf` file.

Running the status tool

The Directory Server provides a **`status`** tool that outputs the current state of the server as well as other information, such as server version, java runtime environment statistics, operation processing times, work queue, and administrative alerts.

About this task

The **`status`** tool is located in the `bin` directory for UNIX and Linux or the `bat` directory for Windows.

Steps

1. Run the **`status`** command on the command line.

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

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

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

          --- Server Details ---
Host Name:            server1.example.com
Administrative Users: cn=Directory Manager
Installation Path:    PingDirectory
Server Version:       PingDirectory Server
                  8.1.0.0
Java Version:         jdk-7u9

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

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

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

           --- Operation Processing Time ---
Op Type   : Total Ops : Avg Resp Time (ms)
----------:-----------:--------------------
Add       : 0         : 0.0
Bind      : 0         : 0.0
Compare   : 0         : 0.0
```

```
Delete    : 0         : 0.0
Modify    : 2788567   : 0.921
Modify    : 0         : 0
DN        : 2267266   : 0.242
Search    : 5055833   : 0.616
All

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

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

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

> ⓘ **Note:**
>
> By default, the **`status`** command displays the alerts generated in the last 48 hours. To change this default setting, see step 2.

2. To limit the number of viewable alerts from the default 48 hours, use the `--maxAlerts` option.

Tuning the Server

PingDirectory Server's installation process automatically determines the optimal Java Virtual Machine (JVM) settings based on calculations of the machine running setup.

The default configuration and JVM settings are suitable for most deployments, but it is not uncommon in high-performance environments to make slight changes to the Directory Server's JVM settings as well as performance and resource-related configuration changes with the **`dsconfig`** tool. For these high-performance environments, tuning can achieve optimum throughput performance and disk space usage for PingDirectory Server and its tools.

About minimizing disk access

Minimizing disk access is critical to the directory server performance.

Defining a Java Virtual Machine (JVM) heap size that can contain the entire contents of the database cache in memory minimizes read operations from disk and achieves optimal performance. The database on-disk is comprised of transaction log files, which are only appended to. After an initial database import, the size on-disk will grow by a factor of at least 25% as inactive records accumulate in the transaction logs. During normal operation, the on-disk size of the database transaction logs does not represent the memory needed to cache the database.

Consider minimizing the size of the database based on the known characteristics of your data. Doing so reduces hard disk requirements and the memory requirements for the database cache. An example of this is the Directory Server automatically compacting common parent distinguished names (DN).

Finally, consider the write load on your server and its effect on the database. Write operations will always require an associated write-to-disk, but an environment that sustains a high load of write operations might consider tuning the background database cleaner to minimize the size of the database on disk.

Memory allocation and database cache

Memory allocation and database cache comprise the basic components of the Directory Server footprint and logic behind the automated tuning of the **`setup`** tool.

The Directory Server's optimal performance is dependent on:

- The proper allocation of memory to the Java Virtual Machine (JVM) heap
- The number of processor cores in the system
- The correct combination of JVM options for optimized garbage collection

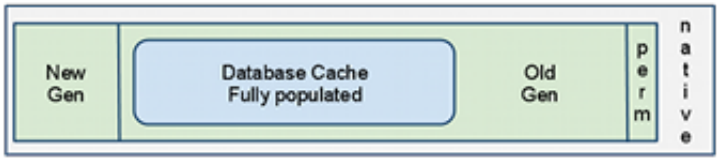
The **`setup`** tool for Directory Server automatically assigns the JVM options and determines the memory allocation based on the total amount of memory on the system; however, in most production deployments, additional tuning might be required to meet the performance objectives for your system.

Often directory server performance tuning can be accomplished by adjusting a few settings. Tuning these settings, which include both JVM and configuration options, require an understanding of the JVM heap structure and the expected database usage.

Directory Server process memory

The Directory Server consists mainly of a Java Virtual Machine (JVM) heap and a marginal amount of memory allocated by the JVM's execution of native code.

While we frequently refer to the JVM Heap as the maximum memory consumed by the Directory Server, the actual process size is slightly larger than the Xmx value because of accumulation of small chunks of native code that Java requires for things, such as SSL sockets.



MY TITLE Java Development Kit (JDK) Heap Structure

Within the JVM Heap, the principal memory components are the new and old generations. The new generation is a smaller area of memory where all data is initially allocated and is cleaned of garbage often. Any data that is present long enough is promoted to the old generation for the longer term. The old generation is where the database cache eventually resides. The old generation size is computed from the leftover heap after defining the `MaxHeapSize` and new generation sizes It is not explicitly stated in the JVM options. A typical set of generation definitions for the JVM is as follows, where mx and ms values represent the heap size.

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

> ⓘ **Note:**
>
> The mx and ms values should always be the same, and the MaxNewSize and NewSize values should be the same. This helps avoid negative changes in performance.

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

Determining heap and database cache size

Define proper memory allocation of the directory server components using the Directory Server **`setup`** command.

About this task

To define the proper memory allocation of the directory server components:

Steps

1. Run the Directory Server **`setup`** command.

   > ⓘ **Note:**
   >
   > This must be done on hardware that represents the target production platform, especially with regard to process and memory, and the largest heap size that the **`setup`** tool allows.

2. To define schema and production database settings for the database import, use the **`import-ldif`** tool.

   > ⓘ **Note:**
   >
   > After running **`import-ldif`**, the database is at its most optimized state on-disk with no inactive records. Over time, the on-disk representation of the database grows up to 25-50% as inactive records accumulate before being removed by the server's cleaner thread.

3. After the database is imported, start the server and make any needed configuration changes.
   Set the prime-method to preload on the `userRoot` backend configuration.
4. Restart the Directory Server and watch for a successful preload message.
   - If preloading completes, proceed to step 6.
   - If preloading does not complete, proceed to step 5.
5. If preloading does not complete, troubleshoot with the following steps:
   a. In the **`config/java.properties`** file, in the **`start-server.java-args`** entry, edit the entry to use larger values for **`-Xmx`** and **`-Xms`** arguments.
   b. Run the following command.

      ```
      bin/dsjavaproperties
      ```

   c. Restart the server and proceed to step 6.
6. After preload completes, run the **`status`** command to review the database cache utilization.

   > ⓘ **Note:**
   >
   > A fully loaded database needs at least 10-20% cache headroom available for future growth, as in the following example.

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

7. Optional: To see the state of the database cache in more detail, run an **`ldapsearch`** on the backend monitor.

> ⓘ **Note:**
>
> In addition to the user configured backends, there might be backends for replication and changelog. The heap is shared among all backends. For information on how the heap amount allocated to each backend is calculated, see*Automatic DB cache percentages* on page 285.

Automatic DB cache percentages
Adjust the allocation of the DB cache percentage when setting up the server.

About this task

The setup process automatically tunes the percentage of the `db-cache-percent` property for the `userRoot` backend based on the maximum configured Java Virtual Machine (JVM) heap size. This is only done for the `userRoot` backend during setup. Other backends created by the user are allocated 10%. Change the allocation if needed. When setting up the server:

Steps

1. Install the server with the necessary memory.

> ⓘ **Note:**
>
> The server autotunes the size of the cache.

2. Set the autotuned cache size to the limit for the combined cache sizes of all of the backends.
3. Divide the server cache based on the expected size of the data in each backend.

Automatic memory allocation
If the Memory Tuning feature is enabled during setup, the **`setup`** algorithm determines the maximum Java Virtual Machine (JVM) heap size based on the total amount of available system memory. Otherwise, the server allocates a maximum JVM heap of 384 MB.

About this task

Directory Server also allows you to specify the maximum heap size during the setup process. For more information, see *JVM Properties for Server and Command-Line Tools*.

Steps

- To enable Memory Tuning during the setup process:
  - Select the feature in interactive command-line mode.
  - Add the `--jvmTuningParameter` option using the **`setup`** tool in non-interactive command-line mode.
  - Regenerate the Java properties file with **`bin/dsjavaproperties`** and the `--jvmTuningParameter` options.

> ⓘ **Note:**
>
> If Memory Tuning is selected, the server allocates the maximum JVM heap depending on the total system memory. The following table displays the automatically allocated maximum JVM heap memory based on available system memory.

Allocated Max JVM Memory if Tuning is Enabled

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

Automatic memory allocation for the command-line tools
At setup, the Directory Server automatically allocates memory to each command-line utility based on the maximum Java Virtual Machine (JVM) heap size.

The server sets each command-line utility in the `config/java.properties` file with -Xmx/Xms values, depending on the expected memory needs of the tools.

Because some tools can be invoked as a server task while the server is online, there are two definitions of the tool in the `config/java.properties` file:

`.Online`

Typically requires minimal memory because the task is performed within the Directory Server's JVM.

`.Offline`

Can require the same amount of memory needed by the Directory Server. Examples include `import-ldif.offline` and `rebuild-index.offline`.

With large databases, some tools, such as **`ldap-diff`** and **`verify-index`**, might need more than the minimal memory. The following table lists the tools that are expected to have more than the minimal memory needs along with the rules for defining the default heap size.

Default Memory Allocation to the Command-Line Tools

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

Database preloading

The ability to maintain the database contents in the database cache within the Java Virtual Machine (JVM) memory is key to the Directory Server performance.

With a properly sized database cache, a priming method of preload directs the server to load the database contents into memory at server startup before accepting the first client connection. The time needed to preload the database is proportional to the database size. To avoid priming, you can start the server with the **`start-server --skipPrime`** command. If the priming method is `none`, or the **`--skipPrime`** option is specified at startup, the database cache slowly builds as entries are accessed. This can take several days to reach optimal performance.

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

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

The following is a summary of the priming methods:

Preload All Data

Prime the contents of the backend into the database cache.

Preload Internal Nodes Only

Prime only internal database structure information into the database cache, but do not prime any actual data. This corresponds to the cache-keys-only cache-mode.

Cursor Across Indexes

Use the **`cursor-across-indexes`** property to iterate through backend contents. This is similar to and might be slower than using the preload mechanism, but it enables priming to happen in the background after the server has started. This is used when shorter start up times are desired, and the slower performance of an uncached database is acceptable until the database is primed.

Configuring database preloading
Use the **dsconfig** tool to set the database priming method.

If multiple prime methods are used, the order in which they are specified in the configuration is the order in which they are performed. Changing the preloading option requires restarting the Directory Server. The following procedures shows how to configure database preloading.

Configuring database preloading
Configure database preloading.

Steps

1. To load the database contents from disk into memory when the server starts up, set the prime method to preload.

   This eliminates the need for the server to gradually prime the database cache using client traffic and ensures that the server has optimal performance when it starts to receive client connections.

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

2. To apply the changes, restart the Directory Server.
   a. Run **bin/stop-server**.
   b. Run **bin/start-server**.

Configuring multiple preloading methods
Configure multiple preloading methods.

Steps

1. To achieve the benefits of preloading without delaying server startup, set **prime-method** to **preload-internal-nodes-only**, which caches all of the keys within the database, but not the values.

   The database values themselves can be cached in the background once the server has been started with the **cursor-across-indexes** option.

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

2. To apply the changes, restart the Directory Server.
   a. Run **bin/stop-server**.
   b. Run **bin/start-server**.

Configuring system index preloading
Configure system index preloading.

About this task
Some environments have many indexes configured though only a few are used for performance-sensitive traffic.

Steps

1. To reduce server start up time, preload only the necessary indexes into the database at startup.

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

```
  --set prime-all-indexes:false \
  --set system-index-to-prime:dn2id \
  --set system-index-to-prime:id2entry
```

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

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

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

2. To apply the changes, restart the Directory Server.
   a. Run **`bin/stop-server`**.
   b. Run **`bin/start-server`**.

Databases on storage area networks, network-attached storage, or running in virtualized environments

There are several considerations when using network-based storage or storage abstracted by virtualization that are not issues when databases are stored on local disks.

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

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

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

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

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

Database cleaner

Production environments that have a high volume of write operations might require cleaner thread tuning to control the on-disk database size as log files with inactive nodes wait to be cleaned and deleted.

The Directory Server stores its Oracle Berkeley DB Java Edition (JE) database files on-disk in the `db` directory. Each JE database log file is labeled `nnnnnnnn.jdb`, where `nnnnnnnn` is an 8-digit hexadecimal number that starts at 00000000 and is increased by 1 for each file written to disk. JE only appends data to the end of each file and does not overwrite any existing data. JE uses one or more cleaner threads that run in the background to compact the number of JE database (db) files.

The cleaner threads begin by scanning the records in each db file, starting with the file that contains the smallest number of active records. Next, the cleaner threads append any active records to the most recent database file. If a record is no longer active due to modifications or deletions, the cleaner threads leave it untouched. After the db file no longer has active records, the cleaner threads can either delete the file or rename the discarded file.

> ⓘ **Note:**
>
> Because of this approach to cleaning, the database size on-disk can temporarily increase when cleaning is being performed and files are waiting to be removed.

The local DB backend configuration object has two properties that control database cleaning: `db-cleaner-min-utilization` and `db-num-cleaner-threads`. The `db-cleaner-min-utilization` property determines, by percentage, when to begin cleaning out inactive records from the database files. By default, the property is set to 75, which indicates that database cleaning ensures that at least 75% of the total log file space is devoted to live data.

> ⓘ **Note:**
>
> This property only affects the on-disk representation of the database and not the in-memory database cache—only live data is ever cached in memory.

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

If the number of database files grow beyond your expected guidelines or if the Directory Server is experiencing an increased number of update requests, you can increase the number of cleaner threads using the **`dsconfig`** tool by going to **Backend**# **Select Advanced Properties**# **db-num-cleaner-threads**.

Compacting common parent DNs

PingDirectory Server compacts entry distinguished names (DNs) by tokenizing common parent DNs.

About this task

Tokenizing the common parent DNs allows you to increase space usage efficiency when encoding entries for storage. The Directory Server automatically defines tokens for base DNs for the backend, such as `dc=example,dc=com`. You can also define additional common base DNs you want to tokenize.

Steps

- Use the following configuration to tokenize two branches: `ou=people,dc=example,dc=com` and `ou=customers,dc=example,dc=com`.

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

Setting the import thread count

For most systems, the default setting of 16 threads is sufficient and provides good import performance. On some systems, increasing the import thread count can lead to improved import performance while selecting a value that is too large can actually cause import performance to degrade.

About this task

If minimizing LDIF import time is crucial to your deployment, you must determine the optimal number of import threads for your system, which is dependent on both the underlying system and the data set being imported.

Steps

- Use the **dsconfig** command to set the number of import threads.

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

JVM properties for server and command-line tools

The Directory Server and tools refer to the `config/java.properties` file for Java Virtual Machine (JVM) options that include important memory settings.

The `java.properties` file sets the default Java arguments for the Directory Server and each command-line utility includes the default *<JAVA_HOME>* path.

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

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

Applying changes using dsjavaproperties

To apply the changes to the `config/java.properties` file, edit the file manually, and then run the **bin/dsjavaproperties** utility.

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

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

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

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

Updating the Java version in the properties file
To change the version of Java that is used by the server and tools, edit the `config/java.properties` file and apply the change by invoking **`bin/dsjavaproperties`** with no command line options.

About this task

You must restart the server for the change to take affect.

Steps

- Inside **`config/java.properties`**, alter the value of **`default.java-home`** to point to the Java correct Java Runtime Environment (JRE).

  Any time the **`config/java.properties`** file is updated, the **`bin/dsjavaproperties`** tool must be run to apply the new configuration.

```
$ bin/dsjavaproperties
```

Regenerating the Java properties file
The **`dsjavaproperties`** command provides an `--initialize` option that allows you to regenerate the Java properties file specifically if you set up the Directory Server using standard memory usage but opt for aggressive memory tuning after setup.

About this task

Rather than reconfigure the Java properties file by re-running **`setup`** or manually editing the `java.properties` file, you can regenerate the properties file for aggressive memory tuning. Any existing file is renamed with a `.old` suffix.

Steps

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

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

Tuning for disk-bound deployments

Configure the server for a disk-bound configuration.

About this task

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

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

Steps

1. When installing the server, choose the **Aggressive** option for JVM memory configuration and to preload the data when the server starts.
2. Set the `default-cache-mode` of the `userRoot` backend to `cache-keys-only`.

3. Set operating system `vm.swappiness` to 0 to protect the Directory Server JVM process from an overly aggressive file system cache.
4. When the data set is imported with the above settings, verify in the `import-ldif` output that the cached portions of the data set fit comfortably within the database cache.

Uncached attributes and entries

Although achieving optimal Directory Server performance requires that the entire data set be fully cached, there can be deployments in which fully caching the data set is not possible due to hardware or financial constraints, or in which acceptable performance can be achieved by only caching a portion of the data.

The Directory Server already provides support for controlling caching on a per-database basis, such as to cache only certain indexes and system databases, but these features might not provide sufficient control over how memory is used, particularly with regard to which entries are included in the cache, and they do not provide any degree of control over caching only a portion of attributes.

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

The `uncached-id2entry` database can be included in the set of databases to prime, but if priming is to be performed, it only includes internal nodes and not leaf nodes. For example, the internal nodes of the `uncached-id2entry` database are included in the preload if the `prime-all-indexes` option is set to "true," or if the `system-index-to-prime-internal-nodes-only` option has a value of `uncached-id2entry`.

Backup and Restore

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

Replication

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

LDIF Import and Export

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

Server Access Log

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

Uncached Entry and Attribute Properties

The Directory Server provides three new advanced properties on the local DB backend to control the caching mode for the `uncached-id2entry` database:

uncached-id2entry-cache-mode

Specifies the cache mode that is used when accessing the records in the `uncached-id2entry` database. If the system has enough memory available to fully cache the internal nodes for this database, then `cache-keys-only` is recommended. Otherwise it is better to select `no-caching` to minimize the amount of memory required for interacting with the `uncached-id2entry` database. For more information, see the PingDirectory Server Configuration Reference.

uncached-attribute-criteria

Specifies the criteria used to identify attributes that are written into the `uncached-id2entry` database, rather than the `id2entry` database. This property is only used for entries in which the associated `uncached-entry-criteria` does not indicate that the entire entry should be uncached. The property applies to all entry writes, including add, soft delete, modify, and modify DN operations, as well as LDIF import and re-encode processing. Any changes to the property take effect immediately for writes occurring after the change is made. If no value is specified, then all attributes are written into the `id2entry` database.

uncached-entry-criteria

Specifies the criteria used to identify entries that are written into the `uncached-id2entry` database, rather than the `id2entry` database. The property applies to all entry writes, including add, soft delete, modify, and modify DN operations, as well as LDIF import and re-encode processing. Any changes to the property take effect immediately for writes occurring after the change is made. If no value is specified, then all entries are written into the `id2entry` database.

Configuring uncached attributes and entries
Configure uncached attributes and entries.

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

Steps

1. Run **`dsconfig`** to uncache entries that match the criteria.

```
$ bin/dsconfig create-uncached-entry-criteria \
  --criteria-name "Fully Uncached l=austin" --type filter-based \
```

```
  --set enabled:true --set "filter:(l=austin)"
```

The filter uncaches all entries that have its location set to "austin", such as `l=austin`.

2. Run **`dsconfig`** to uncache attributes that match the criteria.

   The `--type simple` option indicates that the simple uncached attribute criteria be used to specify the attribute-type that should be uncached. For those entries that are fully stored in the `uncached-id2entry` database container, the uncached attribute is ignored.

   In this example, the attribute-type criteria that should be uncached is `jpegPhoto`.

```
$ bin/dsconfig create-uncached-attribute-criteria \
  --criteria-name "Uncached jpegPhoto" --type simple \
  --set enabled:true --set attribute-type:jpegPhoto
```

3. Set the uncached properties for the `userRoot` backend.

```
$ bin/dsconfig set-backend-prop \
  --backend-name userRoot \
  --set "uncached-entry-criteria:Fully Uncached l=austin" \
  --set "uncached-attribute-criteria:Uncached jpegPhoto"
```

4. Run the **`re-encode-entries`** tool to initiate a task that causes a local DB `userRoot` backend to re-encode all or a specified subset of the entries that it contains.

   The tool does not alter the entries themselves but provides a useful mechanism for applying significant changes to the way that entries are stored in the backend.

   The following example initiates a task that re-encodes all fully-cached entries in the `userRoot` backend, rate-limited to no more than 100 entries per second.

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

JVM garbage collection using CMS

To ensure reliable server performance with Java, the Directory Server depends on Java's Concurrent Mark and Sweep process (CMS) for background garbage collection.

Out of several garbage collection options, CMS is the ideal choice for consistent system availability. For the most part, the CMS collector runs as one or more background threads within the Java virtual machine (JVM), freeing up space in JVM Heap from an area called the old generation. One of the criteria used by CMS to determine when to start background garbage collection is a parameter called CMSInitiatingOccupancyFraction. This percentage value, which applies to the Old Generation, is recommended for the JVM to initiate CMS when data occupancy in Old Generation reaches the threshold.

Ideally, the database cache takes less than 70% of the space available in the Old Generation, and the `CMSInitiatingOccupancyFraction` value of 80 leaves plenty of headroom to prevent the JVM from running out of space in Old Generation because of an inability for CMS to keep up. Because CMS takes processing resources away from the Directory Server, do not set the CMSInitiatingOccupancyFraction at or below the expected database cache size, which would result in the constant running of CMS in the background. For a description of determining Old Generation size, see the section on *Memory Footprint and Database Cache*.

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

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

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

Also, when you start the server, an administrative alert indicates the current state of the CMSInitiatingOccupancyFraction and its recommended value.

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

... (more output) ...

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

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

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

> ⓘ **Note:**
>
> The generated value for the Directory Server property could change over time with each Directory Server build, Java release, or changes in data set. If the current value is close to the recommended value, then you do not need to change the property unless the server experiences a JVM pause.

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

Determining the CMSInitiatingOccupancyFraction
Use the terminal to determine the CMSInitiatingOccupancyFraction value.

Steps

1. Retrieve the `prime-method` property for the backend.

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

> **Note:**
>
> If you set the `Preload Database at startup` option during the installation, then skip to step 3.

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

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

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

At startup, you see an administrative message if the current CMSInitiatingOccupancyFraction property is below the recommended value.

> **Note:**
>
> You can get the recommended value from this message and change it in the `config/java.properties` file in step 5.

3. If you were unable to see the recommended CMSInitiatingOccupancyFraction property at startup presented in the previous step, pre-tune the value of the CMSInitiatingOccupancyFraction property to ensure that all of the data is imported into the server and preloading is enabled in the backend.
   a. Retrieve the recommended CMSInitiatingOccupancyFraction value by issuing the following search.

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

> **Note:**
>
> If the `recommended-cms-initiating-occupancy-fraction-for-current-data-set` is not present, then make sure that the server has been restarted since enabling preload for the backends.

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

4. Open the `config/java.properties` file using a text editor and manually edit the CMSInitiatingOccupancyFraction (or any other property) to its recommended value in the `start-server.java-args` property.
5. Save the file.

   The following arguments are recommended for a Sun 5440 server. Contact your authorized support provider for specific assistance.

   ```
   start-server.java-args=-d64 -server -Xmx20g -Xms20g -XX:MaxNewSize=1g
   -XX:NewSize=1g -XXParallelGCThreads=16 -XX:+UseConcMarkSweepGC
   -XX:+CMSConcurrentMTEnabled -XX:+CMSParallelRemarkEnabled
   -XX:+CMSParallelSurvivorRemarkEnabled -XX:ParallelCMSThreads=8
   -XX:CMSMaxAbortablePrecleanTime=3600000
   -XX:+CMSScavengeBeforeRemark -XX:RefDiscoveryPolicy=1
   -XX:CMSInitiatingOccupancyFraction=55 -XX:+UseParNewGC
   -XX:+UseBiasedLocking -XX:+UseLargePages
   -XX:+HeapDumpOnOutOfMemoryError
   ```

   The `-XX:ParallelGCThreads` should be limited to 16 (default) or to 8 for smaller systems. The `-XX:ParallelCMSThreads` should be limited to 8.
6. To apply the changes, run the **`bin/dsjavaproperties`** command.

   ```
   $ bin/dsjavaproperties
   ```
7. Restart the Directory Server.

Configuring the Server

Configure the server to meet the performance, hardware, operating system, and memory requirements for your production environment.

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

The Directory Server stores its configuration settings in an LDIF file, `config/config.ldif`. Rather than editing the file directly, the Directory Server provides command-line options and an administrative console to configure the server. The Directory Server also includes tools to create server groups to apply configuration changes to multiple servers at one time

About the configuration tools

Use the administrative console or the `dsconfig` command-line tool to modify the PingDirectory Server.

You can access and modify the PingDirectory Server configuration in the following ways:

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

About the dsconfig configuration tool

The **dsconfig** tool is the text-based management tool used to configure the underlying Directory Server configuration.

The `dsconfig` tool has three operational modes: interactive mode, non-interactive mode, and batch mode.

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

Using dsconfig in interactive command-line mode

In interactive mode, the **dsconfig** tool offers a filtering mechanism that only displays the most common configuration elements.

About this task

The user can specify that more expert level objects and configuration properties be shown using the menu system.

Running **dsconfig** in interactive command-line mode provides a user-friendly, menu-driven interface for accessing and configuring the PingDirectory Server.

Steps

1. To start `dsconfig` in interactive command-line mode, invoke the `dsconfig` script without any arguments.
   You are prompted for connection and authentication information to the Directory Server, and then a menu displays the available operation types.
2. To accept the default values, press **Enter**.

> ⓘ **Note:**
>
> In some cases, a default value is provided in square brackets. For example, [389] indicates that the default value for that field is port 389.

3. To skip the connection and authentication prompts, provide the connection and authentication information using the command-line options of `dsconfig`.

Configuring the Server using dsconfig interactive mode
Configure the PingDirectory Server using the `dsconfig` command-line tool in interactive mode.

Steps

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

```
$ bin/dsconfig
```

2. Enter the LDAP connection parameters.
   - Enter the Directory Server host name or IP address.
   - Press **Enter** to accept the default.
3. Enter the number corresponding to the type of LDAP connection that you are using on the Directory Server, or press **Enter** to accept the default.

   The numbers for the LDAP connection type are 1 for LDAP, 2 for SSL, and 3 for StartTLS. The default entry is 1.

4. Enter the LDAP listener port number, or accept the default port.

   The default port is the port number of the server local to the tool.
5. Enter the user bind DN and the bind DN password.

   The default is `cn=Directory Manager`.
6. On the **Directory Server Configuration Console** main menu, enter a number corresponding to the configuration that you want to change.

   > ⓘ **Note:**
   >
   > The number might change between releases or within the same release, depending on the options selected, such as in cases where more expert level objects and properties are displayed.
7. In this example, select the number for Backend.
8. Set the `db-cache-percent` to 40%.

   The optimal cache percentage depends on your system performance objectives and must be tuned as determined through analysis. In many cases, the default value chosen by the **`setup`** utility is sufficient.
9. On the **Backend management** menu, enter the number corresponding to view and edit an existing backend.
10. Select the backend to work with or press **Enter** to accept the default.

    In this example, using the basic object menu, only one backend that can be viewed in the directory, `userRoot`.
11. From the **Local DB Backend** properties menu, type the number corresponding to the `db-cache-percent` property.
12. Enter the option to change the value, and then type the value for the `db-cache-percent` property.

    In this example, type `40` for 40 %.
13. To apply the changes, enter `f`.

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

Viewing dsconfig advanced properties

Configure **`dsconfig`** interactive mode to hide or display additional advanced properties.

About this task

> ⓘ **Note:**
>
> For most configuration settings, some properties are more likely to be modified than others.

Steps

1. Repeat steps 1–9 in *Configuring the Server using dsconfig interactive mode* on page 299.
2. To display the advanced properties, from the **Local DB Backend properties** menu, enter `a`.
   This toggles any hidden properties.

Changing the dsconfig object menu

The purpose of object levels is topresent only those properties that an administrator will likely use.

About this task

Because some configuration objects are more likely to be modified than others, the PingDirectory Server provides four different object menus that hide or expose configuration objects to the user. The `object` type is a convenience feature designed to improve menu readability.

The following object menus are available:

Basic

Only includes the components that are configured most frequently.

Standard

Includes all components in the Basic menu plus other components that might occasionally need to be altered in many environments.

Advanced

Includes all components in the Basic and Standard menus plus other components that might require configuration under special circumstances or that might be harmful if configured incorrectly.

Expert

Includes all components in the Basic, Standard, and Advanced menus plus other components that almost never require configuration, or that could seriously impact the functionality of the server if not properly configured.

To change the `dsconfig` object menu:

Steps

1. Using `dsconfig`, repeat steps 1–6 in *To Install the in Interactive Mode*.
2. On the **PingDirectory Server configuration** main menu, enter the letter `o` to change the object level.

   Basic objects are displayed by default.
3. Enter a number corresponding to an object level of your choice.
   - Enter `1` for Basic.
   - Enter `2` for Standard.
   - Enter `3` for Advanced.
   - Enter `4` for Expert.
4. Review the menu at the new object level.

   Additional configuration options for the Directory Server components are displayed.

dsconfig interactive administrative alerts

The **`dsconfig`** tool and the administrative console provide a feature that displays notifications for certain operations that require further administrative action to complete the process.

If you change a certain backend configuration property, the admin action appears in two places during a **`dsconfig`** interactive session:

- When configuring the property
- Before you apply the change

For example, if you change the `db-directory` property on the userRoot backend, such as specifying the path to the file system path that holds the Oracle Berkeley DB Java Edition backend files, an admin action reminder displays during one of the steps.

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

```
Enter choice [b]: f One or more configuration property changes require
 administrative action or confirmation/notification. Those properties include:
 * db-directory: Modification requires that the Directory Server be stopped,
 the database directory manually relocated, and then the Directory Server
 restarted. While the Directory Server is stopped, the directory and files
 pertaining to this backend in the old database directory must be manually
 moved or copied to the new location. Continue? Choose 'no' to return to the
 previous step (yes / no) [yes]:
```

```
Enter choice [b]: f One or more configuration property changes require
administrative action or confirmation/notification. Those properties include:
* db-directory: Modification requires that the Directory Server be stopped,
the database directory manually relocated, and then the Directory Server
restarted. While the Directory Server is stopped, the directory and files
pertaining to this backend in the old database directory must be manually
moved or copied to the new location. Continue? Choose 'no' to return to the
previous step (yes / no) [yes]:
```

Currently, only a small set of properties that display an admin action alert appear in **dsconfig** interactive mode and the administrative console. For more information on the properties, see the *PingDirectory Server Configuration Reference*, located in the server's `docs/config-guide` directory.

Using dsconfig in non-interactive mode

The **dsconfig** non-interactive command-line mode provides a simple way to make arbitrary changes to the Directory Server by invoking it from the command line.

Steps

1. To use administrative scripts to automate configuration changes, run the **dsconfig** command in non-interactive mode.

   Non-interactive mode is convenient for scripting applications.

   > ⓘ **Note:**
   >
   > If you plan to make changes to multiple configuration objects at the same time, then the batch mode might be more appropriate.

2. Use the **dsconfig** tool to update a single configuration object using command-line arguments to provide all of the necessary information.

   The following shows the general format for the non-interactive command line.

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

   > ⓘ **Note:**
   >
   > The `--no-prompt` argument indicates that you want to use non-interactive mode. The `{sub-command}` is used to indicate which general action to perform. The `{globalArgs}` argument provides a set of arguments that specify how to connect and authenticate to the Directory Server. Global arguments can be standard LDAP connection parameters or SASL connection parameters depending

on your setup. For example, using standard LDAP connections, you can invoke the **dsconfig** tool, as shown.

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

3. If your system uses SASL GSSAPI (Kerberos), invoke **dsconfig** as shown.

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

4. To always display the advanced properties, use the `--advanced` command-line option.

> ⓘ **Note:**
>
> The `{subcommandArgs}` argument contains a set of arguments specific to the particular subcommand that you want to invoke.
>
> Global arguments can appear anywhere on the command line, including before the subcommand and after or intermingled with subcommand-specific arguments. The subcommand-specific arguments can appear anywhere after the subcommand.

Configuring the Server using dsconfig non-interactive mode

Use `dsconfig` non-interactive mode to modify memory and specify parent distinguished names (DNs).

Steps

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

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

Viewing a list of dsconfig properties

Run `dsconfig` commands to view a list of properties, objects, property descriptions, and property information.

Steps

1. To view the list of all `dsconfig` properties, run the **dsconfig** command with the `list-properties` option.

> ⓘ **Note:**
>
> Remember to add the LDAP connection parameters.

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

2. To view objects at and below the menu object level, use the **dsconfig** command with the `list-properties` option and the `--complexity <menu level>` option.

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

> ⓘ **Note:**
>
> You can also add the `--includeDescription` argument that includes a synopsis and description of each property in the output. Remember to add the LDAP connection parameters.

3. If the server is offline, you can run the command with the `--offline` option without needing to enter the LDAP connection parameters.

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

4. To review the property information provided with the server, view the *<server-root>*`/docs/config-properties.txt` file.

Getting the equivalent dsconfig non-interactive mode command

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

About this task

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

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

To get the equivalent `dsconfig` non-interactive mode command:

Steps

1. Use **dsconfig** in interactive mode to make changes to a configuration, but do not apply the changes (that is, do not enter the letter `f`).
2. To view the equivalent non-interactive command, enter `d`.
3. View the equivalent command, and then press **Enter** to continue.

   Based on an example in the previous section, changes made to the `db-cache-percent` returns the following.

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

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

Using dsconfig batch mode

Configure the Directory Proxy Server in `dsconfig` batch mode.

About this task

The PingDirectoryProxy Server provides a **dsconfig** batching mechanism that reads multiple **dsconfig** invocations from a file and executes them sequentially.

The batch file provides advantages over standard scripting by minimizing LDAP connections and Java virtual machine (JVM) invocations that normally occur with each **dsconfig** call. Batch mode is the best method to use with setup scripts when moving from a development environment to test environment or from a test environment to a production environment. The `--no-prompt` option is required with **dsconfig** in batch mode.

If a **dsconfig** command has a missing or incorrect argument, the command fails and aborts the batch process without applying any changes to the Directory Proxy Server. The **dsconfig** command supports a **--batch-continue-on-error** option that instructs **dsconfig** to apply all changes and skip any errors.

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

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

Steps

1. Create a text file that lists each **dsconfig** command with the complete set of properties that you want to apply to the Directory Proxy Server.

> ⓘ **Note:**
>
> The items in this file should be in the same format as those accepted by the **dsconfig** command.
>
> The batch file can have blank lines for spacing and lines starting with a pound sign (#) for comments.
>
> The batch file also supports a "\" line continuation character for long commands that require multiple lines.

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

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

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

# Note that once the entry-count placement algorithm is created
# and enabled, we can delete the round-robin algorithm.
# Since an entry-balancing proxy must always have a placement
# algorithm, we add a second algorithm and then delete the
```

```
# original round-robin algorithm created during the setup
# procedure.

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

2. To read and execute the commands, run **dsconfig** with the `--batch-file` option.

Using PingDirectory Server or PingDirectoryProxy Server with PingFederate OAuth tokens

Configure an access token validator to use PingFederate OAuth tokens with PingDirectory.

Before you begin

You need the following information:

- The runtime engine service port of the PingFederate server, usually 9031
- The client ID of an OAuth 2.0 client configured on the PingFederate server
- The client secret of the OAuth 2.0 client
- The IP address of the PingFederate server

About this task

After you configure a PingFederate server to issue OAuth 2 tokens, you must make these tokens compatible with SCIM 2.0 operations on PingDirectory Server or PingDirectoryProxy Server.

This section explains how to set up an access token validator to handle this task.

Steps

1. Register the PingFederate server using the following command.

```
dsconfig create-external-server \
  --server-name PingFederateInstance \
  --type http \
  --set base-url:https://<PingFed IP address>:<PingFed port> \
  --set hostname-verification-method:allow-all \
  --set "trust-manager-provider:Blind Trust"
```

> ⓘ **Note:**
>
> In this example, the hostname verification method is set to allow-all and the Blind Trust manager provider is used for the sake of simplicity. You should not use these settings for production environments.

2. Create the access token validator using the following command.

```
dsconfig create-access-token-validator \
  --validator-name PingFederateValidator \
  --type ping-federate \
  --set enabled:true \
  --set authorization-server:PingFederateInstance \
  --set client-id:client-id \
```

```
  --set client-secret:client-secret
```

> ⓘ **Note:**
>
> Take the *client-id* and *client-secret* values from the PingFederate OAuth 2 client that will be used with the PingDirectory Server or PingDirectoryProxy Server.

3. Add the access token validator to the SCIM 2 HTTP Servlet configuration with the following command.

```
dsconfig set-http-servlet-extension-prop \
  --extension-name SCIM2 \
  --set access-token-validator:PingFederateValidator
```

4. Test the validator by sending a GET request to `/scim/v2/ServiceProviderConfig`.

   This endpoint does not require any scopes to access, just a valid bearer token. The sever should return a response similar to the following.

```
{
    "schemas": [
        "urn:ietf:params:scim:schemas:core:2.0:ServiceProviderConfig"
    ],
    "patch": {
        "supported": true
    },
    "bulk": {
        "supported": false,
        "maxOperations": 0,
        "maxPayloadSize": 0
    },
    "filter": {
        "supported": true,
        "maxResults": 0
    },
    "changePassword": {
        "supported": true
    },
    "sort": {
        "supported": false
    },
    "etag": {
        "supported": false
    },
    "authenticationSchemes": [
        {
            "name": "OAuth 2.0 Bearer Token",
            "description": "The OAuth 2.0 Bearer Token Authentication
 scheme. OAuth enables clients to access protected resources by obtaining
 an access token, which is defined in RFC 6750 as \"a string representing
 an access authorization issued to the client\", rather than using the
 resource owner's credentials directly.",
            "specUri": "http://tools.ietf.org/html/rfc6750",
            "type": "oauthbearertoken",
            "primary": true
        }
    ],
    "meta": {
        "resourceType": "ServiceProviderConfig",
        "location": "https://localhost:8443/scim/v2/ServiceProviderConfig"
    }
}
```

About recurring tasks and task chains

You can use the **dsconfig create-recurring-task** command to create recurring tasks and task chains to perform regular maintenance tasks for PingDirectory Server. These tasks can perform regular backups, LDIF exports, enter and exit lockdown mode, or other static operations.

Because this process is owned by the server, tasks do not require special privileges or credentials, and they can be run when the server is offline.

Create tasks and then add them to a recurring task chain for scheduling. The task chain ensures that invocations of a task or set of tasks run in a specified order and do not overlap.

A recurring task includes:

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

After you create a task, add one or more tasks to a task chain and schedule the chain using the **dsconfig create-recurring-task-chain** command.

A recurring task chain includes:

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

> ⓘ **Note:**
>
> Changing the schedule for an existing recurring task chain only takes effect the next time an instance of the chain needs to be scheduled. It does not affect any existing instances of the chain that are already scheduled. Existing scheduled instances still run at the originally scheduled time. When that run is complete, the server schedules the next iteration according to the then-current schedule logic.
>
> Also, you cannot simply cancel the existing scheduled instance to make the next instance run earlier. When you cancel an instance of a recurring task chain, the server automatically schedules the next instance for the next time that matches the scheduling criteria. It never schedules a new instance for earlier than or the same time as one that you manually canceled.

Creating a recurring task and task chain

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

Steps

1. To create a task, use **dsconfig create-recurring-task**.

   The following creates a backup task.

```
$ bin/dsconfig create-recurring-task \
  --task-name backup-1 \
  --type backup \
  --set 'email-on-failure:admin1@company.com' \
  --set 'email-on-failure:admin2@company.com' \
  --set compress:true \
  --set encrypt:true \
  --set "retain-previous-full-backup-age:4 w" \
  --set retain-previous-full-backup-count:10
```

2. To create a task chain to schedule and run recurring tasks, use **`dsconfig create-recurring-task-chain`**.

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

LDIF export as a recurring task
You can create a recurring task to schedule the export of LDIF data.

For new installations, the server exports LDIF data by default every day at 1:05 a.m. in the default time zone for the Java virtual machine (JVM), which is generally the time zone configured for the underlying system. At the scheduled time, the server exports the contents of each public backend to a file in the server root `ldif` directory.

> ⓘ **Note:**
>
> Public backends are non-administrative backends containing user-supplied data.

The server compresses and encrypts the LDIF exports if the global configuration is set to encrypt LDIF exports by default, which is enabled if encryption is configured during setup. The LDIF exports are rate limited to ten megabytes per second to minimize the impact on server performance, and exports are retained for seven days.

The recurring task chain is created in instances that are updated to this release. However, the task chain is not enabled by default.

LDIF export can export multiple backends in the same recurring task. You can use the **`backend-id`** property to include multiple backends. You can use the **`exclude-backend-id`** property to exclude one or more backends. These optional properties are mutually exclusive:

- If the **`backend-id`** property has one or more values, only the backends with those IDs are exported.
- If the **`exclude-backend-id`** property has one or more values, all public backends except those listed are exported.
- If neither the **`backend-id`** property nor the **`exclude-backend-id`** property supply values, all public backends are exported.

Lockdown mode as a recurring task
You can create recurring tasks to move a server in and out of lockdown mode. Moving a server in and out of lockdown mode is useful for scheduling other tasks while the server is mostly idle and not accepting connections from clients.

While in lockdown mode, the server reports itself as unavailable and also rejects requests from any user that doesn't have the **`lockdown-mode`** privilege.

The recommended flow for a recurring task chain that uses lockdown mode would be:

1. Enter lockdown mode task.
2. Delay task that waits for the work queue to report that the server is idle.
3. Run desired tasks to perform while the server is in lockdown mode.
4. Leave lockdown mode task.

The enter lockdown mode and leave lockdown mode tasks each allow one optional property that you can use to supply a description for why that the server is being moved into this mode.

File retention recurring task
You can configure a recurring task to remove files in a specified directory that match a given pattern.

You can exclude files that match count-based, time-based, or space-based retention criteria. If any files are to be removed, the oldest files are removed before the most recent file.

If the file name pattern includes a "`${timestamp}`" element, the timestamp is used to identify the file's age. If the file name pattern does not include a timestamp, then the file's age is determined by using the file's creation time, if available, or the last modified time, if the creation time is not available. If a file's age cannot be determined, the file is not removed.

If multiple files have the same age, the server uses lexicographic ordering to differentiate between them. Lexicographic ordering is applicable only for files with no **`retain-file-age`** property configured or for files that are older than that age. If multiple files do not share the same age, but that age is younger than the **`retain-file-age`** value, then those files are retained.

When configuring a file removal task, you must specify at least one of the following properties:

- **`retain-file-count`**
- **`retain-file-age`**
- **`retain-aggregate-file-size`**

Using exec tasks

Exec tasks allow administrators and external users to execute a specified command on the server once or as recurring tasks.

About this task

The server restricts the kinds of commands that can be executed, and the access level of users who can execute them.

These safeguards and requirements include:

- The absolute path to the command to execute must be listed in the *<server-root>*`/config/exec-command-whitelist.txt` file.
- The global configuration must be updated to allow the exec task. The server does not permit it by default. The following command enables this.

```
$ bin/dsconfig set-global-configuration-prop \
  --add allowed-task:com.unboundid.directory.server.tasks.ExecTask
```

- The user scheduling the task must have the **`exec-task`** privilege. The server does not grant permission to run this task to any user by default, including root users.

The following configuration changes grant the **`exec-task`** privilege to a single root user, all root users, or a single non-root user:

Steps

- To grant the **`exec-task`** privilege to a single root user, run the following.

```
$ bin/dsconfig set-root-dn-user-prop --user-name "<username>" \
  --add privilege:exec-task
```

- To grant the **`exec-task`** privilege to all root users, run the following.

```
$ bin/dsconfig set-root-dn-prop \
  --add default-root-privilege-name:exec-task
```

- To grant the **`exec-task`** privilege to a single non-root user, run the following.

```
dn: <userdn>
```

```
changetype: modify
add: ds-privilege-name
ds-privilege-name: exec-task
```

- Use the **schedule-exec-task** tool to create an exec task from the command line.

  The following command schedules an exec task to run the **verify-index** tool to check the integrity of the **cn** index in the backend that hosts **"dc=example,dc=com"**, assuming that the server is installed in **/ds**.

```
$ bin/schedule-exec-task --hostname directory.example.com \
  --port 389 \
  --bindDN uid=admin,dc=example,dc=com \
  --promptForBindPassword \
  --waitForCompletion \
  --logCommandOutput \
  /ds/bin/verify-index --baseDN dc=example,dc=com --index cn
```

Topology configuration

Topology configuration enables automatic server grouping and configuration change mirroring. It uses a master and slave architecture for mirroring shared data across the topology.

All writes and updates are forwarded to the master, which forwards them to all other servers. Reads can be served by any server in the group, and servers can be added to an existing topology at installation.

> ⓘ **Note:**
>
> To remove a server from the topology, you must uninstall it with the uninstall tool.

Topology master requirements and selection

A topology master server receives configuration changes from other servers in the topology, verifies the changes, and then makes the changes available to all connected servers.

When updating, the master sends a digest of its subtree contents. If the node's digest differs from the master's, the server node knows it is not synchronized. The servers pull the entire subtree from the master if they detect that they are not synchronized.

A server detects it is not synchronized with the master under the following conditions:

- The server's subtree digest differs from the master's digest at the end of its periodic polling interval.
- One or more servers have been added to or removed from the topology.

The master of the topology is selected by prioritizing servers based on:

- Minimum supported product version
- Availability
- Server version
- Earliest start time
- Startup UUID (smaller is preferred)

After determining a master, the topology data is reviewed from all available servers, every five seconds by default, to determine if any new information identifies a better server master. If a new server can be the master and no other servers have indicated that they should be the master, it will communicate its eligibility to the other servers. This ensures that all servers accept the same master at approximately the same time, within a few milliseconds of each other. If there is no better master, the initial master maintains the role.

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

- A majority of servers is reachable from that master.

  The master server itself is considered while determining this majority.

- There is only a single master in the entire topology.

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

Topology components
When you install a server , you can add it to an existing topology, cloning the server's configuration. Topology settings are designed to operate without additional configuration. If required, some settings can be adjusted to fit the needs of the environment.

Server configuration settings

Configuration settings for the topology are configured in the global configuration and in the config file handler backend. They are topology settings, but they are unique to each server and are not mirrored. Settings must be kept the same on all servers.

The global configuration object contains a single topology setting, force-as-master-for-mirrored-data. This should be set to `true` on only one of the servers in the topology, and is used only if the topology cannot determine a master because most of the servers are not available. A server with this setting enabled is assigned the role of master if no suitable master can be determined.

The config file handler backend defines three topology **`mirrored-subtree`** settings:

mirrored-subtree-peer-polling-interval

Specifies the frequency at which the server polls its topology peers to identify any changes warranting a new master selection. A lower value ensures a faster failover, but it also causes more traffic among the peers. The default value is 5 seconds. If no suitable master is found, the polling frequency adjusts to 100 milliseconds until a new master is selected.

mirrored-subtree-entry-update-timeout

Specifies the maximum length of time to wait for an entry update operation, such as add, delete, modify or modify-dn, to be applied by the master on all of the servers in the topology. The default is 10 seconds, but updates can take up to twice as long if master selection is in progress at the time the update operation is received.

mirrored-subtree-search-timeout

Specifies the maximum length of time in milliseconds to wait for search operations to complete. The default is 10 seconds.

Topology settings

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

Monitor data for the topology
Each server has a monitor that exposes that server's view of the topology in its monitor backend, so that peer servers can periodically read this information to identify changes in the topology.

Topology data includes the following:

- The server ID of the current master, if the master is not known
- The instance name of the current master or if a master is not set, a description stating why a master is not set
- A flag indicating which server thinks that it should be the master

- A flag indicating which server is the current master
- A flag indicating a server that was forced as master
- The total number of configured peers in the topology group
- The peers connected to this server
- The current availability of this server
- A flag indicating that a server is not synchronized with its master or another node in the topology if the master is unknown
- The amount of time in milliseconds that multiple masters were detected by this server
- The amount of time in milliseconds that no suitable server is found to act as master
- A SHA-256 digest encoded as a base-64 string for the current subtree contents

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

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

Servers and certificates

The following provides a detailed description of PingDirectory Server certificate types.

PingDirectory Server uses two main types of certificates:

Listener certificates

Used to secure communication through TLS.

Inter-server certificates

Used to authenticate between server instances in a topology.

Listener certificates

When a client initiates TLS negotiation with the server, the server presents a certificate chain to the client and the certificate at the head of the chain functions as a listener certificate.

Because the client decides whether to trust the certificate chain, it is recommended that the chain be signed by an issuer whom the client is likely to trust or that the client can be easily configured to trust.

You can create self-signed certificates with long lifespans, but a certificate that a certification authority signs is likely to have a relatively short lifespan. Commercial authorities typically issue certificates that are valid for only one or two years, but some authorities use shorter validity windows.

Short certificate lifespans offer some security benefits. In particular, because most clients do not verify whether a certificate has been revoked, a shorter validity window minimizes the timeframe that a compromised certificate can be used. If the process for replacing certificates is streamlined or automated, administrative inconvenience can be kept to a minimum.

Listener certificates are stored in key stores that are referenced by key manager providers, which in turn provide the logic and configuration for accessing the key stores. If a server component, like a connection handler, requires access to a certificate that it presents to a peer during the TLS negotiation process, that component must reference the key manager provider that points to the key store containing the appropriate certificate. If the key store contains multiple certificates, and if the component referencing the key store includes a property specifying the certificate's nickname, the certificate with that alias is selected. Otherwise, the server lets the Java virtual machine (JVM) select a certificate that might not be well-defined.

The server also provides trust manager providers, which determine whether to trust the certificate chains with which it is presented. A trust manager provider can reference a specified trust store file, but other

options include the JVM default trust store, which uses the Java installation's default set of trusted issuers, and the blind trust manager provider, which automatically trusts every certificate chain that is presented to it.

> ⓘ **Note:**
>
> Never use a blind trust manager in a production environment because it leaves the server vulnerable to impersonation and man-in-the-middle attacks. However, a blind trust manager can be convenient in test environments when troubleshooting certain types of problems.

Replacing listener certificates

Certificate authorities typically restrict the lifespans of the certificates that they sign. If you use a certification authority to issue listener certificates, you are likely replacing the certificates on a regular basis.

About this task

The **`replace-certificate`** tool performs the following steps:

1. Obtain a new certificate chain.
2. Make necessary updates to the key manager provider and the connection handler configurations
3. Update the server instance listener configuration with the new certificate.

The **`replace-certificate`** tool offers the following modes of operation:

Interactive mode

Walks you through the process of obtaining a new certificate and installing it in the server. Interactive mode also displays the non-interactive commands that are required to achieve the same result.

Non-interactive mode

Useful when scripting the process of replacing a certificate.

Steps

- To replace a listener certificate, run the **`replace-listener-certificate`** subcommand of the **`replace-certificate`** tool.

> ⓘ **Note:**
>
> You can replace certificates manually, but the **`replace-certificate`** tool automates the process. The **`replace-certificate`** tool provides information about multiple listener certificates during the transitional phase that occurs when you install them.
>
> The **`replace-listener-certificate`** subcommand takes arguments that provide the following information:
>
> - Arguments required to authenticate to PingDirectory Server, such as --bindDN and --bindPasswordFile
> - Details about the key store that contains the new certificate
> - Updates that must be made to the key and trust manager providers
> - Whether to signal the HTTP connection handler to reload its certificates after the update is complete

The following arguments are available:

Argument	Description
--source-key-store-file {path}	Path to the Java KeyStore (JKS) or PKCS #12 file that contains the private key entry with the new certificate chain. This argument is required.
--source-key-store-password {password}	Clear-text password that is needed to access the contents of the source key store.
--source-key-store-password-file {path}	Path to the file that contains the password necessary to access the contents of the source key store. The file can contain the password in the clear or can be encrypted with a definition from the server's encryption settings database.
--source-certificate-alias {alias}	Password that is required to access the appropriate private key in the source key store. If neither the --source-private-key-password nor the --source-private-key-password-file argument is provided, the key store password is used as the private key password.
--source-private-key-password-file {path}	Path to the file that contains the password needed to access the appropriate private key in the source key store. The file can contain the password in the clear or can be encrypted with a definition from the server's encryption settings database. If neither the --source-private-key-password nor the --source-private-key-password-file argument is provided, the key store password is used as the private key password.
--key-manager-provider {name}	Name of the key manager provider that is updated to use the new certificate chain. The value must identify a file-based key manager provider, and the new certificate chain must be enabled. Defaults to JKS if a value is not specified.

Argument	Description
--trust-manager-provider {name}	Name of the trust manager provider that is updated with the information required to trust the new certificate chain. The value must identify a file-based trust manager provider, and the new certificate chain must be enabled. If neither this argument nor the --use-jvm-default-trust-manager-provider argument is provided, the tool assumes that the name of the trust manager provider is identical to the name of the key manager provider.
--use-jvm-default-trust-manager-provider	Indicates that the server must be configured to use the JVM-default trust manager provider, which trusts certificates signed by issuers in the `cacerts` trust store provided with the Java virtual machine (JVM), rather than updating an existing trust manager provider.
--target-certificate-alias {alias}	Alias to use for the new certificate in the key manager provider's key store, and for appropriate updates in the trust manager provider's trust store. Defaults to an alias of server-cert if a value is not specified. ⓘ **Note:** If the key manager provider's key store, or the trust manager provider's trust store, already contains an entry with the given alias, the existing entry is renamed.
--reload-http-connection-handler-certificates	Indicates that the tool is requesting that the server cause any HTTPS-based connection handlers to reload their certificates, so that the connection handlers can use the updated certificate. LDAP connection handlers react to the change immediately and start presenting the new certificate chain during subsequent TLS negotiations. HTTPS connection handlers continue using the former certificate until the connection handler is restarted or until the connection handler is asked specifically to reload its certificates. ⓘ **Note:** This option might prevent clients with existing TLS sessions that were negotiated with the former certificate from being resumed.

- To remove earlier certificates from the server instance listener configuration, run the purge-retired-listener-certificates subcommand.

> ⓘ **Note:**
>
> The purge-retired-listener-certificates subcommand does not take arguments other than the ones that are required to authenticate to the server.
>
> By default, the **`replace-certificate`** tool updates the server instance listener configuration object to include the new listener certificate, and it merges the old and new certificates residing in the configuration object.

Inter-server certificates
Each server instance in a topology has an inter-server certificate that is generated during the setup process.

The inter-server certificate is not exposed to clients, so a trusted issuer does not need to sign it. Instead, the topology registry, representing a mirrored portion of the configuration with information about every PingDirectory Server instance in the environment, contains the information that each instance needs to trust the inter-server certificates for all other instances.

Inter-server certificates can be used to protect certain secrets that are shared among servers within the topology, like the secrets that are used to digitally sign log files, backups, and LDIF exports. Inter-server certificates include the encryption keys that reversible password-storage schemes use.

The inter-server certificate is generated with a long lifespan. Replace it only when you suspect that its private key is compromised.

Replacing the inter-server certificate
During the installation process, the inter-server certificate is generated with a long lifespan and does not require replacement under normal circumstances. You should replace the inter-server certificate only if you suspect that its private key is compromised.

About this task

The inter-server certificate is intended for use only between server instances within the same topology. Because it is not exposed to regular clients, the inter-server certificate does not need to be trusted.

The **`replace-certificate replace-inter-server-certificate`** command performs the following steps:

- Acquires the new inter-server certificate from a provided Java KeyStore (JKS) or PKCS #12 key store
- Makes the necessary updates to the `config/ads-truststore` file in the server key store
- Updates the server instance configuration object to include the new inter-server certificate

> ⓘ **Note:**
>
> To avoid the need to replace the inter-server certificate on a regular basis,use a self-signed certificate with a long lifespan. Each server instance must possess its own, unique inter-server certificate that satisfies the following conditions:
>
> - Uses an RSA key pair
> - Has a minimum key size of 2048 bits
>
> The following types of certificates are not allowed:
>
> - Certificates with an elliptic curve key pair
> - Certificates with an RSA key that is smaller than 2048 bits

Steps

- To replace the inter-server certificate, run the **`replace-inter-server-certificate`** subcommand of the **`replace-certificate`**.

  The **`replace-inter-server-certificate`** subcommand takes a subset of the arguments that are used with the **`replace-listener-certificate`** subcommand, including the following arguments:

  - --source-key-store-file {path}--source-key-store-password {password}
  - --source-key-store-password-file {path}
  - --source-certificate-alias {alias}
  - --source-private-key-password {password}
  - --source-private-key-password-file {path}
- To delete earlier values that are no longer needed, run the **`purge-retired-inter-server-certificates`** subcommand.

> ⓘ **Note:**
>
> By default, the new inter-server certificate is merged with the existing values in the server instance configuration entry.

X.509 certificates
PingDirectory Server supports X.509 certificates, the most common type of certificates. *RFC 5280* describes X.509v3, which provides the current version of the specification.

An X.509v3 certificate includes the following components:

X.509 encoding version

Enables the differentiation between an X.509v3 certificate and one that conforms to an earlier or later version of the specification.

Serial number of the certificate

Integer value that uniquely identifies a certificate as issued by a certification authority.

Subject DN

Distinguished name for the certificate, which often provides details about the context in which the certificate is to be used. For more information, see *Certificate subject DNs* on page 319.

Issuer DN

Distinguished name for the issuer certificate, which is the certificate used to sign the certificate. For a self-signed certificate, this value matches the subject DN.

Validity window

Indicates the timeframe during which the certificate is considered valid. This component includes the following elements:

- notBefore

  Specifies the earliest time at which the certificate is considered valid.
- notAfter

  Specifies the latest time at which the certificate is considered valid.

Public key

Public portion of a pair of cryptographically linked keys. For more information, see *Certificate key pairs* on page 320.

Signature

A type of cryptographic proof that the certificate truly was sent from the issuer and has remained unaltered. A self-signed certificate is signed with its own private key. Otherwise, it is signed with the issuer's private key.

An X.509v3 certificate might also include the following optional components:

Subject unique ID

Uniquely identifies the certificate. This component has been deprecated in favor of the subject key identifier extension, so it is generally omitted from X.509v3 certificates.

Issuer unique ID

Subject unique ID of the issuer certificate, if available. This component has been deprecated in favor of the authority key identifier extension.

Set of extensions

Provides additional context for the certificate and the manner in which it is used. For more information, see *Certificate extensions* on page 320.

Certificate subject DNs

A certificate's subject distinguished name (DN) provides information about how the certificate should be used.

Like an LDAP DN, a certificate's subject DN consists of a comma-delimited series of attribute-value pairs. However, unlike an LDAP DN, the attribute names in a certificate subject DN are typically written in all uppercase characters.

A certificate's subject DN is also referred to as its subject. The following attributes commonly appear in a certificate subject.

Attribute name	Attribute description
`CN`	Common name **Note:** For a listener certificate, the `CN` attribute typically identifies the host name that clients use to access the certificate. However, the subject alternative name extension is recommended more highly for accomplishing the same task. Most certificate subject DNs include at least the `CN` attribute.
`E`	Email address
`OU`	Name of the organizational unit, such as the relevant department
`O`	Name of the organization or company
`L`	Name of the locality, such as the appropriate city
`ST`	Full name of the state or province
`C`	ISO 3166 country code

A certificate subject includes at least one attribute-value pair, and the `CN` attribute is typically present. Other attributes can be omitted, although the `O` and `C` attributes are also common. For example, a listener certificate for a server with an address of `ldap.example.com`, which is run by the US-based company Example Corp, might have a subject of `CN=ldap.example.com,O=Example Corp,C=US`.

Certificate key pairs
Each certificate contains a key pair that consists of two keys that are linked cryptographically. If you encrypt data with one key, the data can be only decrypted with the other key.

Although a key pair can be created easily when both keys are generated simultaneously, the process of deriving one key from the other is extremely difficult, a process categorized in cryptographic terms as computationally infeasible.

When generating a key pair, one key is designated as the public key, and the other key is designated the private key. The public key can be made widely available, but the private key must be kept secret and not shared with anyone.

As long as the secrecy of the private key is maintained, the key pair can be used to perform the following functions:

- Encryption, sometimes referred to as confidentiality

  If someone wants to send you a secret message without anyone else viewing it, the message can be encrypted with your public key. Only you possess the private key, so only you can decrypt the message.
- Digital signatures

  If you encrypt data with your private key, it can be decrypted only with your public key. Because your public key can be made widely available, this encryption method does not actually protect the content. However, digital signatures prove that a message came from you because only your private key could have generated it.

> ⓘ **Note:**
>
> When generating a digital signature, the entire message is generally not encrypted. Only a hash of the message is encrypted, typically by using a digest algorithm like SHA-256.
>
> This approach protects the integrity of a message. A decrypted signature that matches the digest of the original message guarantees that the message came from you and that it has remained unaltered since you signed it.

The following public key algorithms are used primarily in certificates that facilitate TLS communication:

- RSA, which is based on the multiplication of large prime numbers
- EC, which is based on computations that involve special types of elliptical curves

Although RSA is supported more widely than EC, it is slower and requires larger keys to achieve the same level of security. To support legacy clients, you should use an RSA certificate and choose a key size of at least 2,048 bits.

If all of your clients support EC certificates, you should use an EC certificate with a key size of at least 256 bits.

Certificate extensions
Extensions provide additional context for a certificate.

Some of the more common extension types include the following:

Subject key identifier

Holds a unique identifier for the certificate, which is generally derived from the certificate's public key.

Authority key identifier

Holds the subject key identifier for the issuer certificate. This extension type helps to identify the issuer certificate, especially when presented with an incomplete certificate chain.

Subject alternative name

Holds a list of ways that clients are expected to reference a server when establishing a connection to it.

> ⓘ **Note:**
>
> Clients must take this information into account when deciding whether to trust a server's certificate.
>
> The most common types of values include DNS names, IP addresses, and URIs. DNS names must be fully qualified, but can optionally use an asterisk in the leftmost component to match any single name in that component. For example, `*.example.com` could match `www.example.com` or `ldap.example.com`, but would not match `ldap.east.example.com` or `example.com`.

Key usage

Provides information about the manner in which the certificate is expected to be used. The following key usages are allowed:

`digitalSignature`

Indicates that the certificate can be used for digitally signing data, excluding certificates and certificate revocation lists (CRL).

`nonRepudiation`

Indicates that the certificate can be used to prevent denying the authenticity of a message. `nonRepudiation` is also known as `contentCommitment`.

`keyEncipherment`

Indicates that the certificate can be used to protect encryption keys, such as symmetric keys that are derived during TLS key agreement.

`dataEncipherment`

Indicates that the certificate can be used for encrypting data directly.

`keyAgreement`

Indicates that the certificate's public key can be used for key agreement, such as deriving the symmetric key that protects TLS communication.

`keyCertSign`

Indicates that the certificate can act as a certification authority and be used for signing other certificates.

`cRLSign`

Indicates that the certificate can be used to sign CRLs.

`encipherOnly`

When used in conjunction with `keyEncipherment`, indicates that the public key can be used only for encrypting data during key agreement.

`decipherOnly`

When used in conjunction with `keyEncipherment`, indicates that the public key can be used only for decrypting data during key agreement.

Extended key usage

Acts as an alternative to the key usage extension and provides additional high-level functionality. The following extended key usages are allowed:

`serverAuth`

Indicates that the server can present the certificate to the client during TLS negotiation.

`clientAuth`

Indicates that the client can present the certificate to the server during TLS negotiation.

`codeSigning`

Indicates that the certificate can be used to sign source and compiled code.

`emailProtection`

Indicates that the certificate can be used to sign or encrypt email messages.

`timeStamping`

Indicates that the certificate can be used to assert the time that an event occurred.

`ocspSigning`

Indicates that the certificate can be used to sign an online certificate status protocol (OCSP) response.

Basic constraints

Indicates whether the certificate can act as a certification authority and, if so, the maximum number of intermediate certificates that can follow it in a certificate chain.

Certificate chains

A certificate chain is an ordered list of one or more certificates. In such a chain, each subsequent certificate is the issuer of the previous certificate.

During TLS negotiation, the server presents a certificate chain to the client, which determines whether to trust the chain and continue with the negotiation. The client can also present its own certificate chain to the server.

If a certificate is self-signed, its chain contains only that single certificate. If a certificate is signed by a self-signed certificate authority (CA) certificate, such as a root CA, the chain contains two certificates: the server certificate and the CA certificate that follows it. If a single intermediate CA (a CA certificate that is signed by a root CA) is present, the chain contains the server certificate, followed by the intermediate CA, and then the root CA.

Intermediate certificate authorities are useful for security purposes, especially in commercial authorities. If a client trusts a root CA certificate, it is likely to trust anything with that root CA certificate at the base of its chain. Consequently, the root CA certificate must be kept secure.

ⓘ **Note:**

If the root CA certificate is compromised, any certificate that is directly or indirectly signed by it can no longer be trusted.

With intermediate CA certificates, the root certificate can be kept offline in secure storage and used only when a new intermediate CA certificate must be signed. The intermediate CA certificates can be used to sign end-entity certificates, but must be protected to avoid compromising any of the certificates. A compromised certificate must be revoked along with all of the certificates that it signed. In such a scenario, the root CA can be used to sign a new certificate.

ⓘ **Note:**

The certificate chain that the server presents to the client, or that the client presents to the server, during TLS negotiation does not always need to be the complete chain. If the root CA at the end of the chain is widely trusted, the server can assume that the client already has that root CA in its default set of trusted certificates. The server can leave that root CA off the chain with the assumption that the client will retrieve

it from its default trust store. While the same assumption could theoretically be true for intermediate CA certificates, only the root CA certificate is commonly omitted. When a client receives an incomplete chain, the client looks in its default trust store to determine whether the trust store contains the issuer certificate, which it can identify by using properties like the issuer distinguished name (DN) or an authority key identifier extension.

The certificate at the head of a certificate chain, which appears as the first one in the list, is often called the end-entity certificate. If this certificate appears at the head of the chain that a server presents during TLS negotiation, it is referred to as the server certificate. If the certificate appears at the head of a chain that a client presents, it is referred to as a client certificate. The certificate at the end of a complete chain must be a root CA certificate. In the case of a self-signed certificate, the chain contains only a single certificate that serves both roles.

About representing certificates, private keys, and certificate signing requests
X.509 is an encoding format that uses the ASN.1 distinguished encoding rules (DER), which exist in binary format. When writing a certificate to a file, either a raw DER format or a plaintext format called PEM can be used.

PEM encoding consists of a line that contains the text `-----BEGIN CERTIFICATE-----`, followed by a set of lines that contains the base64-encoded representation of the raw DER bytes (typically with no more than 64 characters per line), followed by a line that contains the text `-----END CERTIFICATE-----`.

The X.509 encoding contains a certificate's public key, but not its private key. The PKCS #8 specification in *RFC 5958* describes the encoding for private keys. This approach uses a DER encoding with a PEM variant that instead uses the following header and footer, respectively.

```
 -----BEGIN PRIVATE KEY-----
-----END PRIVATE KEY-----
```

RFC 5958 also describes an encrypted representation of the private key, but that format is currently unsupported.

The PKCS #10 specification in *RFC 2986* describes the CSR format. This format uses a DER encoding with a PEM variant that uses the following header and footer, respectively.

```
-----BEGIN CERTIFICATE REQUEST-----
-----END CERTIFICATE REQUEST-----
```

Some implementations use the following alternate, nonstandard forms.

```
-----BEGIN NEW CERTIFICATE REQUEST-----
-----END NEW CERTIFICATE REQUEST-----
```

Certificate trust
When a server presents its certificate chain to a client during TLS negotiation, the client decides whether to trust the certificate chain and concludes whether it is communicating with a legitimate server instead of an impostor.

If a client is tricked through DNS hijacking into communicating with a rogue application instead of with a legitimate server, the application can steal the client's credentials, or can fool the client into concluding that it has performed an action that it has not performed. If a rogue application acts as a broker between the client and the legitimate server, the client might be unable to detect the change, and the malicious application might be capable of stealing data or altering the communication. Consequently, you should avoid `trust all` or `blind trust` options in a production environment.

When determining whether to trust a server certificate chain, a client performs the following steps.

Processing steps

1. Verifies that it has received the complete certificate chain.

   If a server presents an incomplete chain, the client must ensure that it can complete the chain with information in an explicitly provided trust store or default trust store. If the client cannot complete the certificate chain, the chain is not trusted.
2. Verifies that each subsequent certificate in the chain is the issuer certificate for, and that its private key was used to sign, the certificate that precedes it.

   > ⓘ **Note:**
   >
   > If a certificate chain contains extraneous certificates, or if a subsequent certificate did not issue the certificate that precedes it, the chain is not trusted.
3. Confirms that it has a reason to trust the certificate at the root of the chain.

   > ⓘ **Note:**
   >
   > This step is generally performed by ensuring that the root certificate authority (CA) certificate can be found in either a default trust store or a trust store that is configured for use by the client. If the client has no prior knowledge of the root CA certificate, the chain is not trusted.
4. Verifies that the current time lies within the validity window for each certificate in the chain.

   > ⓘ **Note:**
   >
   > The chain is not trusted under the following conditions:
   >
   > - When the `notBefore` value of any certificate in the chain is later than the current time.
   > - When the `notAfter` value of any certificate in the chain is earlier than the current time.
5. Verifies that the server certificate at the head of the chain is suitable for the server with which the client thinks it is communicating.

   > ⓘ **Note:**
   >
   > The client must verify that the address used to connect to the server matches one of the following values:
   >
   > - The `CN` attribute of the certificate's subject.
   > - One of the values of any subject alternative name extension.

These steps represent a starting point. If necessary, the client can perform additional types of validation. For example, if a root or intermediate certification authority maintains a certificate revocation list (CRL) or supports the online certificate status protocol (OCSP), the client must verify that none of the certificates in the chain has been revoked. The client can also verify that the CA certificates include the basic constraints extension, and that the server certificate does not contain too many levels. Other checks, like those that use certificate policy extensions, can also be performed.

Keystores and truststores

A keystore is a type of database that holds certificates.

The following examples represent the most common forms of keystores:

- File that uses the Java-specific Java KeyStore (JKS) format
- File that uses the standard PKCS #12 format
- Collection of files that holds certificates and private keys, typically in PEM or DER format
- Hardware security module (HSM) that makes the certificate information available through an interface like PKCS #11

PingDirectory Server supports file-based keystores by using the JKS and PKCS #12 formats and by using hardware security modules that are accessible through PKCS #11. The server does not currently support a keystore format that consists of individual certificate and private key files. To import these files into a JKS or PKCS #12 keystore, use the **`manage-certificates`** tool.

A keystore also represents a collection of entries, each of which is identified by a name that an alias calls. Keystores can have the following entry types:

Private key entries

Contain a certificate chain and a private key. When a server accepts a TLS-based connection, it uses a private key entry to obtain the certificate chain that it presents to the client. The server can also use the private key from the same entry to process its key agreement. Similarly, a client uses a private key entry when presenting its own certificate chain to a server.

Trusted certificate entries

Contain a single certificate without a private key. As the name implies, a trusted certificate entry is intended primarily for use when determining whether to trust a certificate chain that is presented during TLS negotiation.

Secret key entries

Contain a secret key only, without an associated certificate. These types of entries are not used for TLS processing. Instead, they hold symmetric encryption keys or other types of secrets.

A password, sometimes called a PIN, protects the contents of a keystore. In some cases, like with JKS keystores, some content might be accessible without a password, and a password might be required only when trying to access private keys or secret keys. In other cases, like with PKCS #12 keystores, you might need a password for any interaction with the keystore.

Additional passwords can further protect private keys. This approach is often the same as with the keystore password, but the password can be different. This tactic is useful when a single keystore is shared for multiple purposes, for example, and when merely having access to the keystore does not guarantee access to all of the data that it contains.

> ⓘ **Note:**
>
> A truststore is another name for a keystore that is intended primarily for use when determining whether to trust a certificate chain that has been presented. Truststores generally contain primarily trusted certificate entries, but that case is not required.

Java runtime environments typically include a default truststore, often `jre/lib/security/cacerts` or `lib/security/cacerts`, that is prepopulated with several widely trusted certification authority certificates. When presented with a certificate that one of these authorities has signed, the default truststore can allow the certificate to be trusted without any additional configuration. When presented with a self-signed certificate, or when presented with a certificate that is signed by an issuer not in the default truststore, such as a private corporate certification authority, a separate truststore is required.

Transport Layer Security (TLS)
TLS describes a mechanism for securely communicating between two parties that might have no prior knowledge of each other.

TLS is the successor to SSL, and the two terms are often used interchangeably, even though such usage might not technically be correct.

> ⓘ **Note:**
>
> SSL remains the more widely recognized term. The abbreviation TLS occasionally generates confusion with the StartTLS extended operation, particularly in LDAP.

TLS provides security in the form of the following main components:

Certificate trust

Is about reassuring a connection-initiating client that it is communicating with the server to which it intended to connect. To ensure that the server shares the same degree of confidence in the identity and legitimacy of the client, it can ask the client to present its own certificate chain. For more information, see *Certificate Trust*.

Cipher selection

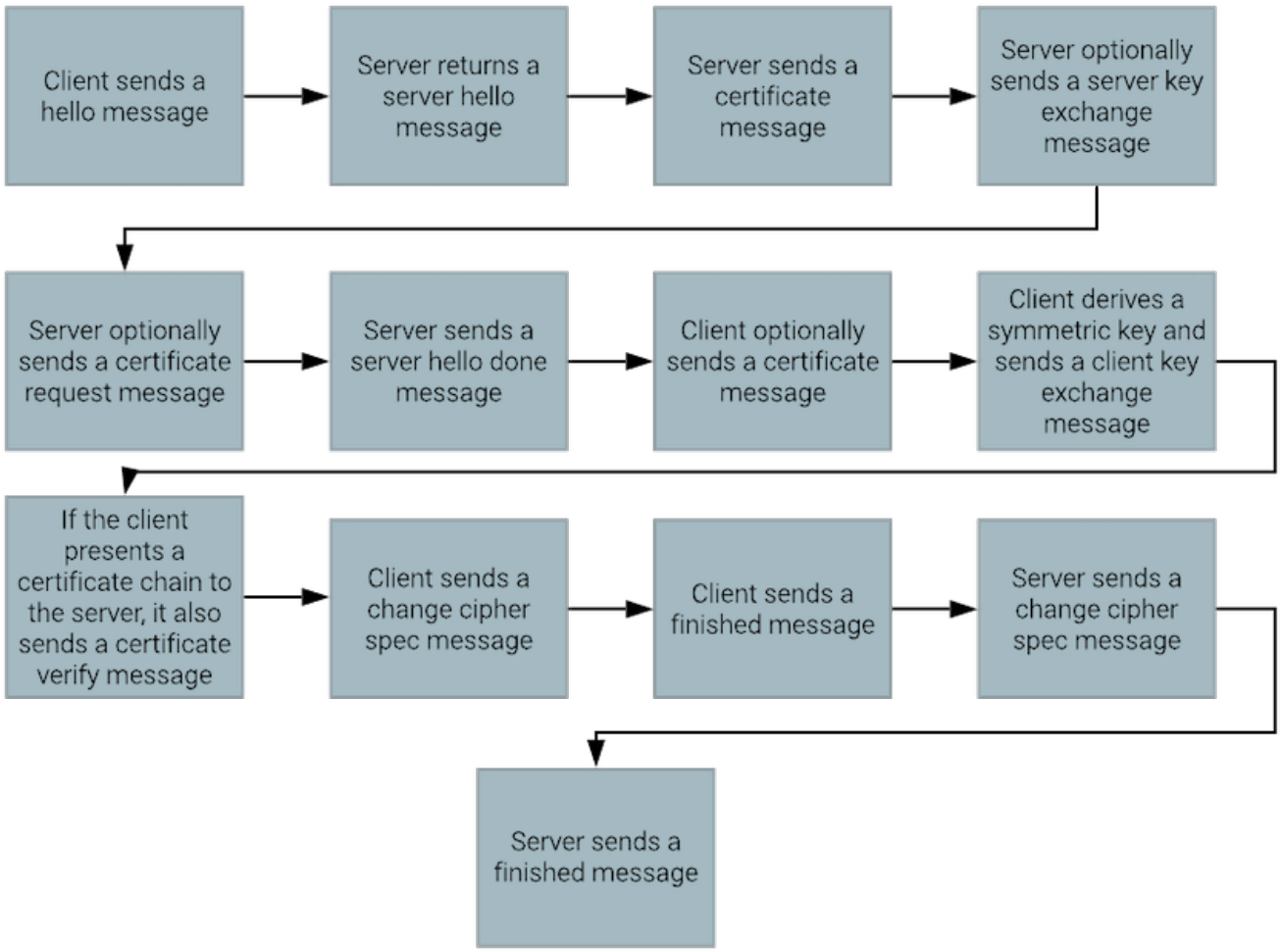
Involves choosing the cipher and the key to protect the bulk of the communication. Although a client can use a server certificate's public key to encrypt data before sending it, this approach can lead to the following issues:

- Unless the client presents its own certificate chain to the server, the server cannot encrypt the data that it sends back to the client.
- Public key encryption is considerably slower than symmetric encryption, in which the same key is used for both encryption and decryption. Public key encryption is also called asymmetric encryption because different keys are used to encrypt and decrypt data.
- If you rely entirely on the security of a private key to ensure the secrecy of a communication, and if the private key becomes compromised, data that has been encrypted with the private key must also be considered compromised.

Rather than relying solely on public key encryption to protect communication between a client and server, the TLS negotiation process allows a client and server to agree on the type of encryption and the secret key to use after completing the negotiation process.

TLS handshakes
The process of negotiating the TLS is referred to as the handshake.

Although the exact process depends on the TLS version that is ultimately chosen, the following steps represent the basic components of a TLS 1.2 handshake:



TLS processing steps

1. The client sends a `hello` message that provides the server with the following information:
   - Highest supported version of the TLS protocol
   - The cipher suites that the client uses
   - Set of extensions with additional information:
     - The address that the client uses to communicate with the server
     - The signature algorithms and elliptic curves that the client supports
     - Whether the client supports secure renegotiation
2. The server returns a `server hello` message that provides the client with the following information:
   - The TLS protocol version that the server uses
   - The cipher suite that the server selects

   The server can also provide its own extensions to the client.
3. The server sends a `certificate` message that provides its certificate chain to the client.
4. The server can optionally send a `server key exchange` message with additional information that the client might need to securely derive the same symmetric encryption key that the server generates.
5. The server can optionally send a `certificate request` message that asks the client to present its own certificate chain to the server.

6. The server sends a `server hello done` message to inform the client that it has completed its `hello` sequence.
7. The client can optionally send a `certificate` message to the server with its own certificate chain.

> ⓘ **Note:**
>
> The client sends a `certificate` message only when the server initially sends a certificate request. If the client receives such a request, it can refuse to, and probably will not, send a certificate chain. The server decides whether to require a client certificate chain. In LDAP, the server commonly asks the client to present a certificate, but continues with TLS negotiation even if the client does not present one. This approach supports authentication methods like SASL EXTERNAL, in which a client uses the certificate chain that it presents during TLS negotiation as proof of its identity.

8. The client derives a symmetric key to use for the remainder of the encrypted processing, and sends a `client key exchange` message to the server. The `client key exchange` message includes the information that the server needs to generate the same key. Only the client and server know the value of the key, even if another entity can observe the communication that passes between the client and the server.
9. If the client presents a certificate chain to the server, it also sends a `certificate verify` message to prove that the private key for the certificate is included at the head of the chain.
10. The client sends a `change cipher spec` message to the server, which informs the server that the client will use the agreed-upon symmetric key to encrypt everything else that it sends to the server.
11. The client sends a `finished` message to the server to indicate that it has completed its portion of the handshake.
12. The server sends a `change cipher spec` message to the client, which informs the client that the server will use the agreed-upon symmetric key to encrypt everything else that it sends to the client.
13. The server sends a `finished` message to the client to indicate that it has completed its portion of the handshake.

TLS 1.3 uses a different handshake sequence that can require only a single round-trip to exchange the necessary information between the client and the server. TLS 1.2 requires two round-trips. To accomplish this task, TLS 1.3 tries to guess the type of key agreement that the server wants to use, and sends the relevant information to the server up front instead of waiting to hear from the server.

Because an extra round of communication between the client and server is eliminated, the server finishes its portion of the negotiation before the client. The server must assume that the client trusts its certificate chain. Because the server might log a successful negotiation only to discover late, through a TLS alert, that the client rejected the certificate, this approach might complicate certain types of troubleshooting.

Key agreement

Key agreement processing provides a critical component of TLS negotiation.

It allows the client and server to select the symmetric key that encrypts the remainder of the communication, but does not reveal the key to anyone who can access the communication. Although several key agreement algorithms are available, the following types are the most common:

RSA key exchange

The client generates random data, uses the server's public key to encrypt it, and provides it to the server, which uses its private key to decrypt it. The client and server alike derive the encryption key from the randomly generated data.

Diffie-Hellman (DH) key exchange

The client and server agree publicly on a pair of mathematically linked numbers, and each participant chooses its own secret value. Through a special computation, they generate a key that can be discovered only by someone who knows one of the secret values. Although several variants of the Diffie-Hellman algorithm can be used in key exchange, we recommend the ECHDE and DHE

versions because they use ephemeral keys with no relation to the server's certificate. Of those two versions, ECDHE is faster and uses smaller keys.

When possible, use ECHDE over DHE, and either of those options over RSA. The DH algorithms provide a substantial benefit over RSA in the form of forward secrecy. Because RSA key exchange uses the server certificate's public key to encrypt data, the encryption can be broken if the certificate's private key is compromised. This warning applies to previously captured data as well as to communication on new TLS connections. The use of ephemeral keys in ECDHE and DHE ensures that, even if the certificate's private key is compromised, the encrypted communication remains indecipherable to anyone but the client and server, although anyone with the private key can still impersonate the legitimate server.

LDAP StartTLS extended operation
In most scenarios, a client that uses TLS establishes a connection to a port that is dedicated to its use, like 636 (LDAPS) or 443 (HTTPS).

The client begins the TLS-negotiation process by sending a `client hello` message over the connection. In some scenarios, the client establishes a non-secure connection and later converts it to a secure one. In LDAP, this task is accomplished by using the **`StartTLS`** extended operation.

The **`StartTLS`** extended operation provides the following advantages over a dedicated LDAPS connection:

- To enable secure as well as insecure communication, only one port needs to be opened through a firewall.
- A client can use opportunistic encryption, in which the client performs the following steps:

  1. Queries the root DSE to determine whether the server supports StartTLS.
  2. Secures the connection, if possible.

  Opportunistic encryption is useful in scenarios like following referrals because LDAP URLs do not officially support LDAPS as a scheme.

To ensure that a communication is always secure, use LDAPS instead of establishing an insecure connection that you secure later with the **`StartTLS`** extended operation. If you enable support for unencrypted LDAP communication, as **`StartTLS`** requires, a client might send a password-containing bind request or other sensitive data over an unencrypted connection. A server can be configured to reject unencrypted communication, but it cannot prevent a client from sending an unencrypted request.

> ⓘ **Note:**
>
> Although you can use **`StartTLS`** to temporarily secure a connection before falling back on an unencrypted LDAP communication, PingDirectory Server does not support this strategy.

About the manage-certificates tool
PingDirectory Server offers a **`manage-certificates`** tool that enables interaction with Java KeyStore (JKS) and PKCS #12 key stores.

Although it behaves similarly to the **`keytool`** utility that accompanies most Java distributions, **`manage-certificates`** is easier to use, provides improved usage information, and offers additional functionality.

Available subcommands

The **`manage-certificates`** tool uses the following subcommands to indicate which function to invoke:

Subcommand	Function
`list-certificates`	Lists the certificates in a keystore.

Subcommand	Function
import-certificate	Imports a certificate into a trusted certificate entry or imports a certificate chain and private key into a private key entry.
export-certificate	Exports a certificate from a keystore.
export-private-key	Exports a private key from a keystore.
generate-self-signed-certificate	Generates a self-signed certificate.
generate-certificate-signing-request	Generates a certificate-signing request that can be provided to a certification authority.
sign-certificate-signing-request	Signs a certificate-signing request with a specified issuer certificate.
check-certificate-usability	Checks a specified certificate in a keystore to verify whether it is suitable for use as a listener certificate.
trust-server-certificate	Initiates the TLS-negotiation process with a specified server to obtain its certificate chain so that a truststore can be updated with the necessary information to trust the chain.
display-certificate-file	Displays the contents of a file that contains one or more PEM-encoded or DER-encoded X.509 certificates.
display-certificate-signing-request-file	Displays the contents of a file that contains a PEM-encoded or DER-encoded PKCS #10 certificate-signing request (CSR).
change-certificate-alias	Changes the alias for an entry in a keystore.
change-keystore-password	Changes the password for a keystore.
change-private-key-password	Changes the password that protects the private key for a specified entry in a keystore.

Common arguments

Most of the **manage-certificates** subcommands require access to a Java KeyStore (JKS) or PKCS #12 keystore. In such instances, use the --keystore argument to specify the path to the keystore.

If the keystore already exists, the tool detects automatically whether it is a JKS or PKCS #12 keystore. If the operation creates a new keystore, you can specify the type explicitly by using the --keystore-type argument, followed by a value of JKS or PKCS12. If you do not specify the keystore type, a default value of JKS is used.

Some situations require you to provide the password that is needed to access the keystore. For a JKS keystore, you might need to provide a keystore password only for operations that involve creating a keystore or accessing a private key. However, you will likely need to provide the password for all operations that involve a PKCS #12 keystore.

To provide a keystore password, use one of the following arguments:

- --keystore-password, followed by the clear-text password for the keystore.

- --keystore-password-file, followed by the path to a file that contains the password for the keystore. The file might contain the password in the clear, or it might be encrypted with a definition from the server's encryption-settings database.
- --prompt-for-keystore-password. If this argument is provided, the tool prompts you interactively to provide the password.

If a private key is protected with a different password than the keystore itself, specify one of the following arguments to provide the private key password:

- --private-key-password, followed by the plaintext password.
- --private-key-password-file, followed by the path to a file that contains the clear-text or encrypted password.
- --prompt-for-private-key-password, which causes the tool to prompt interactively for the password.

Several operations require you to specify the keystore entry to target. In such scenarios, provide the --alias argument, followed by the name of the alias for that entry.

Listing the certificates in a keystore
List the certificates available in a keystore.

Steps

- To list the certificates in a keystore, use the **`list-certificates`** subcommand.

  This subcommand requires you to specify the path to the keystore file, and possibly the password that is needed to access the keystore. The following options are also available:

Option	Description
--alias {alias}	Specifies the alias of the certificate to display. If this value is not provided, all certificates are displayed. To list more than one specific certificate, specify this value multiple times.
--display-pem-certificate	Includes a PEM-encoded representation of each certificate as part of the output.
--verbose	Includes details about each certificate.

  The following command demonstrates the basic listing of a keystore that contains a single certificate chain.

```
$ bin/manage-certificates list-certificates \
     --keystore config/keystore \
     --keystore-password-file config/keystore.pin

Alias:  server-cert (Certificate 1 of 2 in a chain)
Subject DN:  CN=ds1.example.com,O=Example Corp,C=US
Issuer DN:  CN=Example Certification Authority,O=Example Corp,C=US
Validity Start Time:  Saturday, November 9, 2019 at 11:26:09 AM CST
                     (8 minutes, 15 seconds ago)
Validity End Time:  Sunday, November 8, 2020 at 11:26:09 AM CST
(364 days, 23 hours, 51 minutes, 44 seconds from now)
Validity State:  The certificate is currently within the validity window.
Signature Algorithm:  SHA-256 with ECDSA
Public Key Algorithm:  EC (secP256r1)
SHA-1 Fingerprint: 42:f8:85:97:bf:88:bc:74:4b:5b:ce:0c:54:43:9b:44:6b:
                   81:23:a3
SHA-256 Fingerprint: 4f:be:47:ed:36:68:13:38:ba:e8:c0:c5:6c:85:51:97:
                     8b:40:1b:76:10:c0:be:80:15:62:06:96:c5:71:30:df
Private Key Available:  Yes
The certificate has a valid signature.
```

```
Alias:  server-cert (Certificate 2 of 2 in a chain)
Subject DN:  CN=Example Certification Authority,O=Example Corp,C=US
Issuer DN:  CN=Example Certification Authority,O=Example Corp,C=US
Validity Start Time: Saturday, November 9, 2019 at 11:26:08 AM CST
                     (8 minutes, 16 seconds ago)
Validity End Time: Friday, November 4, 2039 at 12:26:08 PM CDT
                   (7299 days, 23 hours, 51 minutes, 43 seconds from now)
Validity State:  The certificate is currently within the validity window.
Signature Algorithm:  SHA-256 with ECDSA
Public Key Algorithm:  EC (secP256r1)
SHA-1 Fingerprint: b8:d0:16:9b:5d:f2:e7:a1:80:79:95:a2:64:b5:aa:ad:80:
                   23:64:16
SHA-256 Fingerprint: cf:98:2a:66:35:6e:6d:f9:5d:25:c6:68:68:04:5a:a8:
                     88:43:ca:b5:c8:e5:c9:95:09:e9:fc:ab:b9:41:ec:71
The certificate has a valid signature.
```

The following sample represents the verbose version of the previous command.

```
$ bin/manage-certificates list-certificates \
     --keystore config/keystore \
     --keystore-password-file config/keystore.pin \
     --verbose

Alias:  server-cert (Certificate 1 of 2 in a chain)
X.509 Certificate Version:  v3
Subject DN:  CN=ds1.example.com,O=Example Corp,C=US
Issuer DN:  CN=Example Certification Authority,O=Example Corp,C=US
Serial Number:  7b:2d:91:6a:ff:51:4f:7a:19:16:26:4f:ce:cb:cb:31
Validity Start Time:  Saturday, November 9, 2019 at 11:26:09 AM CST
(9 minutes, 48 seconds ago)
Validity End Time: Sunday, November 8, 2020 at 11:26:09 AM CST
                   (364 days, 23 hours, 50 minutes, 11 seconds from now)
Validity State:  The certificate is currently within the validity window.
Signature Algorithm:  SHA-256 with ECDSA
Signature Value:
     30:46:02:21:00:cb:d5:5e:45:b2:8a:33:5e:2d:85:23:39:49:d1:3f:8f:dc:
     f8:9e:2f:f3:44:2f:41:0d:69:95:ec:f0:f5:c0:80:02:21:00:ef:8f:32:35:
     3c:88:f4:89:ed:f3:a6:76:
     bb:92:6c:eb:c6:17:ac:61:dc:67:26:f0:ec:67:90:51:28:a1:d0:d5
Public Key Algorithm:  EC (secP256r1)
Elliptic Curve Public Key Is Compressed:  false
Elliptic Curve X-Coordinate:
   -24253153720011259408467676608081666342358203254369897642016197975874105796326
Elliptic Curve Y-Coordinate:
   48722714538591494552787288916186748185323678082126843165293664643134352536146
Certificate Extensions:
     Subject Key Identifier Extension:
          OID:  2.5.29.14
          Is Critical:  false
          Key Identifier:
               21:ad:b9:7a:15:e4:08:13:05:e1:c2:64:0c:86:aa:9b:f0:4c:fb:a0
     Authority Key Identifier Extension:
          OID:  2.5.29.35
          Is Critical:  false
          Key Identifier:
               01:4b:69:99:93:5f:76:51:39:95:61:cc:a9:a8:cb:16:f2:0f:8c:c8
     Subject Alternative Name Extension:
          OID:  2.5.29.17
          Is Critical:  false
          DNS Name:  ds1.example.com
```

```
          DNS Name:  ds.example.com
          DNS Name:  ldap.example.com
          DNS Name:  localhost
          IP Address:  127.0.0.1
          IP Address:  0:0:0:0:0:0:0:1
     Key Usage Extension:
          OID:  2.5.29.15
          Is Critical:  false
          Key Usages:
               Digital Signature
               Key Encipherment
               Key Agreement
     Extended Key Usage Extension:
          OID:  2.5.29.37
          Is Critical:  false
          Key Purpose ID:  TLS Server Authentication
          Key Purpose ID:  TLS Client Authentication
SHA-1 Fingerprint:
   42:f8:85:97:bf:88:bc:74:4b:5b:ce:0c:54:43:9b:44:6b:81:23:a3
SHA-256 Fingerprint:
   4f:be:47:ed:36:68:13:38:ba:e8:c0:c5:6c:85:51:97:8b:40:1b:76:
   10:c0:be:80:15:62:06:96:c5:71:30:df
Private Key Available:  Yes
The certificate has a valid signature.

Alias:  server-cert (Certificate 2 of 2 in a chain)
X.509 Certificate Version:  v3
Subject DN:  CN=Example Certification Authority,O=Example Corp,C=US
Issuer DN:  CN=Example Certification Authority,O=Example Corp,C=US
Serial Number:  43:b7:bb:0c:82:58:42:d8:06:fc:2a:f6:04:e8:2e:8c
Validity Start Time: Saturday, November 9, 2019 at 11:26:08 AM CST
                     (9 minutes, 49 seconds ago)
Validity End Time: Friday, November 4, 2039 at 12:26:08 PM CDT
                   (7299 days, 23 hours, 50 minutes, 10 seconds from now)
Validity State:  The certificate is currently within the validity window.
Signature Algorithm:  SHA-256 with ECDSA
Signature Value:
     30:45:02:21:00:b9:87:50:5d:b7:6a:19:82:99:9b:aa:f1:5d:25:a1:90:3c:
     17:9d:7f:f5:7f:8d:06:b4:57:41:9e:15:c6:5a:af:02:20:0c:00:5e:17:bf:
     ca:bf:0b:ff:db:9f:dc:55:ad:35:eb:df:f6:37:4e:23:83:36:88:d2:cc:
     7d:9e:23:da:78:28
Public Key Algorithm:  EC (secP256r1)
Elliptic Curve Public Key Is Compressed:  false
Elliptic Curve X-Coordinate:

 -20753103001920939059800335367415761738764700353772539765405069978726324039 64
Elliptic Curve Y-Coordinate:

 67079356503908427292378918440889412002659485731683570737365127953554508553 73
Certificate Extensions:
     Subject Key Identifier Extension:
          OID:  2.5.29.14
          Is Critical:  false
          Key Identifier:
               01:4b:69:99:93:5f:76:51:39:95:61:cc:a9:a8:cb:16:f2:0f:8c:c8
     Basic Constraints Extension:
          OID:  2.5.29.19
          Is Critical:  false
          Is CA:  true
          Path Length Constraint:  0
     Key Usage Extension:
          OID:  2.5.29.15
          Is Critical:  false
          Key Usages:
```

```
               Key Cert Sign
               CRL Sign
SHA-1 Fingerprint:
 b8:d0:16:9b:5d:f2:e7:a1:80:79:95:a2:64:b5:aa:ad:80:23:64:16
SHA-256 Fingerprint:

 cf:98:2a:66:35:6e:6d:f9:5d:25:c6:68:68:04:5a:a8:88:43:ca:b5:c8:e5:c9:95:09:
   e9:fc:ab:b9:41:ec:71
The certificate has a valid signature.
```

Generating self-signed certificates

The process of creating a self-signed certificate is straightforward because a self-signed certificate claims itself as its own issuer.

Although self-signed certificates are convenient for testing environments, clients do not trust them by default. Consequently, you should not use them as listener certificates in production environments.

The **`manage-certificates`** tool offers a **`generate-self-signed-certificate`** subcommand that can create a self-signed certificate. In addition to the arguments that provide information about the keystore, certificate alias, and optional private key password, the following arguments are available.

Argument	Description
`--subject-dn {subject}`	Subject DN for the certificate to create. This value is required.
`--days-valid {number}`	Number of days that the certificate remains valid. Defaults to 365 if no value is specified.
`--validity-start-time {timestamp}`	Indicates the time at which the certificate begins its validity window. This value is assumed to reflect the local time zone, and must be expressed in the form YYYYMMDDhhmmss, where a value of 20190102030405 indicates January 2, 2019, at 3:04:05 AM. Defaults to the current time if no value is specified.
`--key-algorithm {name}`	Name of the algorithm to use when generating the key pair. For a listener certificate, this value is typically RSA or EC. Defaults to RSA if no value is specified. **Note:** This argument cannot be used in conjunction with the `--replace-existing-certificate` argument.

Argument	Description
`--key-size-bits {number}`	Length of the key, in bits, to generate. If the `--key-algorithm` argument is given, then `--key-size-bits {number}` must also be specified. Conversely, if the `--replace-existing-certificate` argument is given, then `--key-size-bits {number}` must not be specified. Typical key sizes are: ▪ RSA key – 2048 or 4096 bits If a default RSA key is used but this argument is not provided, a default key size of 2048 bits is used. ▪ Elliptic curve key – 256 or 384 bits
`--signature-algorithm {name}`	Name of the algorithm to use to sign the certificate. If the `--key-algorithm` argument is used to specify an algorithm other than RSA, then `--signature-algorithm {name}` must also be specified. If the `--replace-existing-certificate` argument is used, then `--signature-algorithm {name}` must not be specified. Typical signature algorithms include SHA256withRSA for certificates with RSA keys, and SHA256withECDSA for certificates with elliptic curve keys. If a default key algorithm is used but the `--signature-algorithm {name}` argument is not provided, a default value of SHA256withRSA is used.
`--replace-existing-certificate`	Uses the new certificate to replace an existing certificate in the key store (within the same alias), and reuses the key for that certificate.
`--inherit-extensions`	Indicates that, when replacing an existing certificate, the new certificate contains the same set of extensions as the existing certificate. If the `--replace-existing-certificate` argument is provided, but the `--inherit-extensions` argument is omitted, the new certificate contains only arguments that are provided explicitly.
`--subject-alternative-name-dns {name}`	Indicates that the certificate is expected to have a subject alternative name extension with the provided DNS name. The given name must be fully qualified, although it can contain an asterisk (*) as a wildcard in the leftmost component. To include multiple DNS names in the subject alternative name extension, specify the `--subject-alternative-name-dns {name}` argument multiple times.

Argument	Description
`--subject-alternative-name-ip-address {address}`	Indicates that the certificate is expected to have a subject alternative name extension with the provided IP address. The given address must be a valid IPv4 or IPv6 address. No wildcards are allowed. To include multiple IP addresses in the subject alternative name extension, specify the `--subject-alternative-name-ip-address {address}` argument multiple times.
`--subject-alternative-name-email-address {address}`	Indicates that the certificate is expected to have a subject alternative name extension with the provided email address. To include multiple email addresses in the subject alternative name extension, specify the `--subject-alternative-name-email-address {address}` argument multiple times.
`--subject-alternative-name-uri {uri}`	Indicates that the certificate is expected to have a subject alternative name extension with the provided URI. To include multiple URIs in the subject alternative name extension, specify the `--subject-alternative-name-uri {uri}` argument multiple times.
`--subject-alternative-name-oid {oid}`	Indicates that the certificate is expected to have a subject alternative name extension with the provided object identifier (OID). The given value must be a valid OID. To include multiple OIDs in the subject alternative name extension, specify the `--subject-alternative-name-oid {oid}` argument multiple times.
`--basic-constraints-is-ca {value}`	Indicates that the certificate is expected to have a basic constraints extension, with a specified value of true or false, for the flag indicating whether to consider the certificate a certification authority that can be used to sign other certificates. ▪ For root and intermediate certificate authority (CA) certificates, the `--basic-constraints-is-ca {value}` argument must be present with a value of true. ▪ For end-entity certificates, the `--basic-constraints-is-ca {value}` argument can optionally be present with a value of false. ▪ For a self-signed certificate, specify the `--basic-constraints-is-ca {value}` argument with a value of false to indicate that the certificate is not considered a CA certificate.

Argument	Description
`--basic-constraints-maximum-path-length {number}`	Indicates that the basic constraints extension is expected to include a path length constraint element with the specified value. Use this argument only if `--basic-constraints-is-ca` is provided with a value of true. A path length constraint value of 0 indicates that the certificate can be used to issue only end-entity certificates. A path length constraint value of 1 indicates that the certificate can be used to sign end-entity certificates or intermediate CA certificates, the latter of which can be used to sign only end-entity certificates. A value greater than 1 indicates the presence of several intermediate CA certificates between it and the end-entity certificate at the head of the chain.
`--key-usage {value}`	Indicates that the certificate is expected to have a key usage extension with the specified value. The following values are allowed: ▪ digital-signature ▪ non-repudiation ▪ key-encipherment ▪ data-encipherment ▪ key-agreement ▪ key-cert-sign ▪ crl-sign ▪ encipher-only ▪ decipher-only To include multiple key usages, specify the `--key-usage {value}` argument multiple times.
`--extended-key-usage {value}`	Indicates that the certificate is expected to have an extended key usage extension with the specified value. The following values are allowed: ▪ server-auth ▪ client-auth ▪ code-signing ▪ email-protection ▪ time-stamping ▪ ocsp-signing

For example, the following command can be used to generate a self-signed server certificate.

```
bin/manage-certificates generate-self-signed-certificate \
     --keystore config/keystore \
     --keystore-password-file config/keystore.pin \
     --keystore-type JKS \
     --alias server-cert \
```

```
     --subject-dn "CN=ds.example.com,O=Example Corp,C=US" \
     --key-algorithm EC \
     --key-length-bits 256 \
     --signature-algorithm SHA256withECDSA \
     --subject-alternative-name-dns ds.example.com \
     --subject-alternative-name-dns ds1.example.com \
     --subject-alternative-name-dns localhost \
     --subject-alternative-name-ip-address 1.2.3.4 \
     --subject-alternative-name-ip-address 127.0.0.1 \
     --subject-alternative-name-ip-address 0:0:0:0:0:0:0:1 \
     --key-usage digital-signature \
     --key-usage key-encipherment \
     --key-usage key-agreement \
     --extended-key-usage server-auth \
     --extended-key-usage client-auth

Successfully created a new JKS keystore.

Successfully generated the following self-signed certificate:
Subject DN:  CN=ds.example.com,O=Example Corp,C=US
Issuer DN:  CN=ds.example.com,O=Example Corp,C=US
Validity Start Time: Monday, January 27, 2020 at 03:40:13 PM
 CST
                     (0 seconds ago)
Validity End Time: Tuesday, January 26, 2021 at 03:40:13 PM CST
                   (364 days, 23 hours, 59 minutes, 59 seconds
 from now)
Validity State:  The certificate is currently within the
 validity window.
Signature Algorithm:  SHA-256 with ECDSA
Public Key Algorithm:  EC (secP256r1)
SHA-1 Fingerprint:
 4f:41:82:7f:08:e9:d8:05:8c:19:8b:3e:5b:bc:59:98:d3:15:71:3a
SHA-256 Fingerprint:

 76:e6:8e:c5:c8:8d:27:ce:2b:85:b9:8c:9d:49:3c:06:f4:40:f1:d0:70:67:39:24:fc:
   31:bc:f8:51:83:f2:42
```

Generating certificate signing requests

A certificate signing request (CSR) contains all of the information that a certification authority requires to issue a certificate.

RFC 2986 defines the request format, also known as PKCS #10, and includes the following elements:

- Certificate signing request version
- Requested subject distinguished name (DN) for the certificate
- Public key for the requested certificate
- Requested set of extensions for the certificate
- Signature that proves the requester has the private key for the given public key

To create a certificate signing request, use the **`manage-certificates generate-certificate-signing-request`** command, which performs the following steps:

1. Generated a public and private key pair.
2. Stores the key pair in a key store with a given alias.
3. Outputs the certificate signing request to the terminal.
4. Optionally writes the certificate signing request to a file.

Because a certificate signing request contains many of the same elements as a certificate, the command to generate one takes most of the same arguments as for generating a self-signed certificate. The following arguments are unavailable when generating a CSR:

- `--replace-existing-certificate`
- `--days-valid {number}`
- `--validity-start-time {timestamp}`

The following arguments are available when generating a certificate signing request but not when generating a self-signed certificate:

`--output-file {path}`

Path to a file to which the certificate signing request is written. If this value is not provided, the request is written only to the terminal in PEM form.

`--output-format {value}`

Format to use when writing the certificate signing request. This value can be PEM or DER, but the DER format is used only in conjunction with the `--output-file` argument. Defaults to PEM if the `--output-format {value}` argument is not provided.

`--use-existing-key-pair`

Indicates that the CSR uses a key pair that already exists in the key store with the given alias, rather than generating a new key pair, in which case the specified alias must not already be in use in the key store.

The following example command creates a CSR.

```
bin/manage-certificates generate-certificate-signing-request \
     --output-file ds1-cert.csr \
     --output-format PEM \
     --keystore config/keystore \
     --keystore-password-file config/keystore.pin \
     --keystore-type JKS \
     --alias server-cert \
     --subject-dn "CN=ds.example.com,O=Example Corp,C=US" \
     --key-algorithm EC \
     --key-length-bits 256 \
     --signature-algorithm SHA256withECDSA \
     --subject-alternative-name-dns ds.example.com \
     --subject-alternative-name-dns ds1.example.com \
     --subject-alternative-name-dns localhost \
     --subject-alternative-name-ip-address 1.2.3.4 \
     --subject-alternative-name-ip-address 127.0.0.1 \
     --subject-alternative-name-ip-address 0:0:0:0:0:0:0:1 \
     --key-usage digital-signature \
     --key-usage key-encipherment \
     --key-usage key-agreement \
     --extended-key-usage server-auth \
     --extended-key-usage client-auth

Successfully created a new JKS keystore.

Successfully generated the key pair to use for the certificate signing
request.

Successfully wrote the certificate signing request to file
'/ds/build/package/PingDirectory/ds1-cert.csr'.
```

If the contents of the resulting CSR file are made available to a certification authority to be signed, the resulting signed certificate can be imported into the key store.

To print the contents of a certificate signing request file, use the **`display-certificate-signing-request-file`** subcommand, which supports the following arguments:

--certificate-signing-request-file {path}

Path to the file that contains the certificate signing request to display.

--verbose

Indicates that the command is expected to display verbose information about the request, rather than a basic information set.

The following example demonstrates the basic output from the command.

```
$ bin/manage-certificates display-certificate-signing-request-file \
    --certificate-signing-request-file ds1-cert.csr

    PKCS #10 Certificate Signing Request Version:  v1
    Subject DN:  CN=ds.example.com,O=Example Corp,C=US
    Signature Algorithm:  SHA-256 with ECDSA
    Public Key Algorithm:  EC (secP256r1)
```

The following example demonstrates the verbose output.

```
$ bin/manage-certificates display-certificate-signing-request-file \
     --certificate-signing-request-file ds1-cert.csr \
     --verbose

PKCS #10 Certificate Signing Request Version:  v1
Subject DN:  CN=ds.example.com,O=Example Corp,C=US
Signature Algorithm:  SHA-256 with ECDSA
Signature Value:

 30:45:02:20:46:31:be:9e:6d:2f:0e:e3:d0:80:5c:88:ef:da:86:07:fd:15:b7:62:
     83:45:39:0a:c9:f2:f9:17:eb:08:94:ff:02:21:00:c8:bd:df:57:fa:ea:8c:04:
     df:c5:27:76:e5:b3:3b:4f:df:ec:d3:e4:09:5b:c0:6c:7b:86:39:ec:d0:0e:c1:64
Public Key Algorithm:  EC (secP256r1)
Elliptic Curve Public Key Is Compressed:  false
Elliptic Curve X-Coordinate:
   208628537904757963197889471667098239762296638799662436502070112279302 4
   3221133
Elliptic Curve Y-Coordinate:
   479697739226644990505743464941788269420922508654777168408919906254139
   60212095
Certificate Extensions:
     Subject Key Identifier Extension:
          OID:  2.5.29.14
          Is Critical:  false
          Key Identifier:
               f2:de:fd:bf:d3:2f:96:ef:01:70:2d:0e:85:f5:fb:17:d5:a0:9e:67
     Subject Alternative Name Extension:
          OID:  2.5.29.17
          Is Critical:  false
          DNS Name:  ds.example.com
          DNS Name:  ds1.example.com
          DNS Name:  localhost
          IP Address:  1.2.3.4
          IP Address:  127.0.0.1
          IP Address:  0:0:0:0:0:0:0:1
     Key Usage Extension:
          OID:  2.5.29.15
          Is Critical:  false
          Key Usages:
               Digital Signature
               Key Encipherment
```

```
               Key Agreement
     Extended Key Usage Extension:
          OID:  2.5.29.37
          Is Critical:  false
          Key Purpose ID:  TLS Server Authentication
          Key Purpose ID:  TLS Client Authentication
```

Importing signed and trusted certificates

Use the **`manage-certificates import-certificate`** command to import certificates into a keystore.

This command is used to accomplish the following tasks:

- Import a certificate that a certification authority has signed into the keystore in which the key pair was generated. In this scenario, the certificate is imported into a private key entry and must be imported as a certificate chain rather than an end-entity certificate.
- Import a trusted issuer certificate into a trust store. In this scenario, the certificate is imported into a trusted certificate entry as a single certificate instead of as a chain.
- Import a certificate chain, along with the private key for the end-entity certificate. This approach imports certificates that were generated through another library, like OpenSSL.

In addition to the arguments that provide information about the key store and the alias into which the certificate or certificate chain is imported, the **`manage-certificates import-certificate`** command accepts the following arguments:

`--certificate-file {path}`

Path to the file that contains the certificate to import. The certificate can be in PEM or DER format and can be a single certificate or a certificate chain. If the certificates in the chain reside in separate files, specify the --certificate-file {path} argument multiple times when you import a certificate chain.

`--private-key-file {path}`

Path to the file containing the private key that corresponds to the certificate at the head of the imported chain. The private key can be in PEM or DER format.

`--no-prompt`

Indicates that the certificate is to be imported without prompting for confirmation. By default, a summary of the certificate is displayed, and you must confirm that you want to import it.

The following example command imports a signed certificate into the key store that generates the certificate signing request.

```
$ bin/manage-certificates import-certificate \
     --keystore config/keystore \
     --keystore-password-file config/keystore.pin \
     --alias server-cert \
     --certificate-file ds1-cert.pem \
     --certificate-file ca-cert.pem

The following certificate chain will be imported into the keystore into
 alias
'server-cert', preserving the existing private key associated with that
 alias:

Subject DN:  CN=ds.example.com,O=Example Corp,C=US
Issuer DN:  CN=Example Root CA,O=Example Corp,C=US
Validity Start Time: Sunday, November 10, 2019 at 09:09:23 PM CST
                     (4 minutes, 16 seconds ago)
Validity End Time: Monday, November 9, 2020 at 09:09:23 PM CST
                   (364 days, 23 hours, 55 minutes, 43 seconds from now)
Validity State:  The certificate is currently within the validity window.
```

```
Signature Algorithm:  SHA-256 with ECDSA
Public Key Algorithm:  EC (secP256r1)
SHA-1 Fingerprint:
 02:51:25:43:3e:68:f5:71:36:e3:5d:df:74:de:f6:a1:5a:db:0f:eb
SHA-256 Fingerprint: 1d:b5:eb:3c:f5:ff:bf:79:a2:a5:86:b8:e4:33:76:4d:d7:
                     50:dc:a4:34:95:37:be:89:45:86:1f:5d:79:c3:93

Subject DN:  CN=Example Root CA,O=Example Corp,C=US
Issuer DN:  CN=Example Root CA,O=Example Corp,C=US
Validity Start Time: Sunday, November 10, 2019 at 09:00:07 PM CST
                     (13 minutes, 32 seconds ago)
Validity End Time: Saturday, November 5, 2039 at 10:00:07 PM CDT
                   (7299 days, 23 hours, 46 minutes, 27 seconds from now)
Validity State:  The certificate is currently within the validity window.
Signature Algorithm:  SHA-256 with ECDSA
Public Key Algorithm:  EC (secP384r1)
SHA-1 Fingerprint:
 0e:5c:21:c9:a5:36:0a:24:eb:aa:55:b6:a5:94:0e:e0:56:03:22:e6
SHA-256 Fingerprint:
 77:cf:66:d7:3c:8a:fd:67:2d:b7:36:fd:60:1d:ca:eb:1b:03:b1:
                     12:7b:10:1f:26:05:b7:b9:0d:02:e0:38:3e

Do you want to import this certificate chain into the keystore? yes

Successfully imported the certificate chain.
```

If you do not provide the `--no-prompt` argument, the **`manage-certificates import-certificate`** tool still displays information about the certificates to import. To view additional information about a certificate before you import it, use the **`display-certificate-file`** subcommand, which supports the following arguments:

`--certificate-file {path}`

Path to the file that contains the certificate to view.

`--verbose`

Displays verbose information about the certificate.

The output of the **`display-certificate-file`** subcommand has the same format and content as the **`list-certificates`** subcommand.

Exporting certificates

Use the **`export-certificate`** subcommand to export a single certificate or a certificate chain from a key store to a file in PEM or DER format.

The **`export-certificate`** subcommand supports the normal arguments about the key store and certificate alias, in addition to the following arguments:

`--output-file {path}`

Path to the file to which exported certificates are written. If this value is not provided, the certificates are written to standard output rather than a file.

`--output-format {format}`

Format in which exported certificates are written. The value can be PEM or DER, but the DER format can be used only if the output is written to a file. Defaults to PEM if no value is specified.

`--export-certificate-chain`

Indicates that a certificate chain, rather than the end-entity certificate only, is to be exported.

`--separate-file-per-certificate`

Indicates the use of separate output files for each exported certificate, rather than placing all of the certificates in a single file. If this argument is provided and multiple certificates are to be exported, then .1 is appended to the path for the indicated output file for the first certificate in the chain, .2 is appended for the second certificate, and so on.

The following example exports a certificate chain.

```
$ bin/manage-certificates export-certificate \
     --keystore config/keystore \
     --keystore-password-file config/keystore.pin \
     --alias server-cert \
     --output-file server-cert.pem \
     --output-format PEM \
     --export-certificate-chain \
     --separate-file-per-certificate

Successfully exported the following certificate to '/ds/server-cert.pem.1':
Subject DN:  CN=ds.example.com,O=Example Corp,C=US
Issuer DN:  CN=Example Root CA,O=Example Corp,C=US
Validity Start Time: Sunday, November 10, 2019 at 09:09:23 PM CST
                     (3 hours, 26 minutes, 23 seconds ago)
Validity End Time: Monday, November 9, 2020 at 09:09:23 PM CST
                           (364 days, 20 hours, 33 minutes, 36 seconds from
 now)
Validity State:  The certificate is currently within the validity window.
Signature Algorithm:  SHA-256 with ECDSA
Public Key Algorithm:  EC (secP256r1)
SHA-1 Fingerprint:
 02:51:25:43:3e:68:f5:71:36:e3:5d:df:74:de:f6:a1:5a:db:0f:eb
SHA-256 Fingerprint:
1d:b5:eb:3c:f5:ff:bf:79:a2:a5:86:b8:e4:33:76:4d:d7:50:dc:a4:34:95:37:be:89:45:
86:1f:5d:79:c3:93

Successfully exported the following certificate to '/ds/server-cert.pem.2':
Subject DN:  CN=Example Root CA,O=Example Corp,C=US
Issuer DN:  CN=Example Root CA,O=Example Corp,C=US
Validity Start Time: Sunday, November 10, 2019 at 09:00:07 PM CST
                     (3 hours, 35 minutes, 39 seconds ago)
Validity End Time: Saturday, November 5, 2039 at 10:00:07 PM CDT
                   (7299 days, 20 hours, 24 minutes, 20 seconds from now)
Validity State:  The certificate is currently within the validity window.
Signature Algorithm:  SHA-256 with ECDSA
Public Key Algorithm:  EC (secP384r1)
SHA-1 Fingerprint:
 0e:5c:21:c9:a5:36:0a:24:eb:aa:55:b6:a5:94:0e:e0:56:03:22:e6
SHA-256 Fingerprint:
   77:cf:66:d7:3c:8a:fd:67:2d:b7:36:fd:60:1d:ca:eb:1b:03:b1:12:7b:10:1f:26:
   05:b7:b9:0d:02:e0:38:3e
```

The **`export-certificate`** subcommand exports only the public portion of a certificate. Its private key is not included. To export the private key, use the **`export-private-key`** subcommand, which supports the following arguments, in addition to the usual key store and alias arguments:

`--output-file {path}`

Path to the file to which the exported private key is written. If this value is not provided, the key is written to standard output rather than a file.

`--output-format {format}`

Format in which the exported private key is written. The value can be PEM or DER, but the DER format is used only if the output is written to a file. Defaults to PEM if no value is specified.

The following code provides an example of the **`export-private-key`** subcommand .

```
$ bin/manage-certificates export-private-key \
     --keystore config/keystore \
     --keystore-password-file config/keystore.pin \
     --alias server-cert \
     --output-file server-cert-key.pem \
     --output-format PEM

Successfully exported the private key.
```

Using manage-certificates as a simple certification authority

If your PingDirectory Server instances need to support an arbitrary or unknown set of clients, configure them with certificates from a trusted issuer, such as a commercial authority or the free Let's Encrypt service.

About this task

If you control every client that accesses the servers, you might want to create your own internal certification authority so that you have a common issuer for all servers. In such a scenario, the clients need to trust only the certificates that the issuer signs. Commercial and open-source software packages provide full-featured certification authority functionality, but you can use the **`manage-certificates`** tool to create a certificate authority (CA) certificate that you can use to sign certificate-signing requests.

Steps

1. Create a CA certificate.

   A CA certificate is a self-signed certificate that possesses the following extensions:

   - A key usage extension that includes at least the keyCertSign usage
   - A basic constraints extension that identifies the certificate as a CA certificate

   If you do not plan to use an intermediate CA certificate, the basic constraints extension must have a path length constraint of 0. If you plan to use an intermediate CA certificate, the path length constraint must be 1. Because certificates that the CA certificate signs are valid only for as long as all certificates in the chain remain valid, we recommend that you specify a long lifespan for the CA certificate.

   The following example creates a new root CA certificate.

   ```
   $ bin/manage-certificates generate-self-signed-certificate \
        --keystore /ca/root-ca-keystore \
        --keystore-password-file /ca/root-ca-keystore.pin \
        --keystore-type JKS \
        --alias root-ca-cert \
        --subject-dn "CN=Example Root CA,O=Example Corp,C=US" \
        --days-valid 7300 \
        --key-algorithm RSA \
        --key-size-bits 4096 \
        --signature-algorithm SHA256withRSA \
        --basic-constraints-is-ca true \
        --basic-constraints-maximum-path-length 1 \
        --key-usage key-cert-sign \
        --key-usage crl-sign

   Successfully created a new JKS keystore.

   Successfully generated the following self-signed certificate:
   Subject DN:  CN=Example Root CA,O=Example Corp,C=US
   Issuer DN:  CN=Example Root CA,O=Example Corp,C=US
   Validity Start Time: Monday, January 27, 2020 at 03:47:29 PM CST (0
    seconds ago)
   ```

```
Validity End Time: Sunday, January 22, 2040 at 03:47:29 PM CST
                   (7299 days, 23 hours, 59 minutes, 59 seconds from now)
Validity State:  The certificate is currently within the validity window.
Signature Algorithm: SHA-256 with RSA
Public Key Algorithm: RSA (4096-bit)
SHA-1 Fingerprint:
 bc:8e:5b:30:52:ec:03:63:b4:9a:aa:1a:45:a0:fc:84:49:dd:e8:64
SHA-256 Fingerprint:

 d5:47:06:cd:a2:95:42:61:1f:c7:aa:04:16:1e:c1:70:41:c4:44:48:bf:74:20:5f:1c:
   61:e2:aa:40:08:3a:ff
```

2. Export the public portion of the root CA certificate for future reference.

   When you import a signed certificate, you can import the public portion of the root CA certificate as a standalone certificate into trust stores as well as into part of a certificate chain.

3. Create a new certificate signing request to create an intermediate CA certificate,

   The certificate signing request uses essentially the same settings as the root CA. If you anticipate only a single intermediate CA, its basic constraints extension must have a path length constraint of 0, rather than 1, to indicate that it is used only to sign end-entity certificates and that it cannot create subordinate CA certificates by itself.

   The following example command creates a certificate signing request.

```
$ bin/manage-certificates generate-certificate-signing-request \
     --keystore /ca/intermediate-ca-keystore \
     --keystore-password-file /ca/intermediate-ca-keystore.pin \
     --keystore-type JKS \
     --alias intermediate-ca-cert \
     --subject-dn "CN=Example Intermediate CA,O=Example Corp,C=US" \
     --key-algorithm RSA \
     --key-size-bits 4096 \
     --signature-algorithm SHA256withRSA \
     --basic-constraints-is-ca true \
     --basic-constraints-maximum-path-length 0 \
     --key-usage key-cert-sign \
     --key-usage crl-sign \
     --output-file /ca/intermediate-ca-cert.csr \
     --output-format PEM

Successfully created a new JKS keystore.

Successfully generated the key pair to use for the certificate signing
request.

Successfully wrote the certificate signing request to file
'/ca/intermediate-ca-cert.csr'.
```

4. Use the root CA certificate to sign the certificate signing request for the intermediate CA certificate with the **`sign-certificate-signing-request`** subcommand.

   The **`sign-certificate-signing-request`** subcommand takes most of the same arguments as generating a self-signed certificate. The primary differences between the argument sets are as follows:

   - The key store that contains the certificate uses the provided key store arguments to sign the request. To specify the name of the certificate to use when signing the request, use the `--signing-certificate-alias` argument.
   - To specify the path to the file that contains the certificate signing request file to generate, provide a `--request-input-file` argument.
   - To specify the path to the file to which the signed certificate is written, provide a `--certificate-output-file` argument. If this argument is omitted, the PEM representation of the certificate is written to standard output.
   - To specify the format, PEM or DER, in which the certificate is written to the output file, provide an `--output-format` argument.
   - To specify the subject to use for the signed certificate, use the `--subject-dn` argument. To use the subject DN from the certificate signing request, omit this argument.
   - To specify the name of the signature algorithm, use the `--signature-algorithm` argument.

     > ⓘ **Note:**
     >
     > Because the requester generated the key, you cannot specify the key algorithm or the key length.

   - To indicate that the signed certificate includes every extension that is listed in the certificate signing request, use the `--include-requested-extensions` argument. If this argument is not provided, explicitly specify the set of extensions to include.

   The following example command signs the certificate signing request for an intermediate CA certificate.

   ```
   $ bin/manage-certificates sign-certificate-signing-request \
        --keystore /ca/root-ca-keystore \
        --keystore-password-file /ca/root-ca-keystore.pin \
        --signing-certificate-alias root-ca-cert \
        --days-valid 7300 \
        --include-requested-extensions \
        --request-input-file /ca/intermediate-ca-cert.csr \
        --certificate-output-file /ca/intermediate-ca-cert.pem \
        --output-format PEM

   Read the following certificate signing request:

   PKCS #10 Certificate Signing Request Version:  v1
   Subject DN:  CN=Example Intermediate CA,O=Example Corp,C=US
   Signature Algorithm:  SHA-256 with RSA
   Public Key Algorithm:  RSA (4096-bit)

   Do you really want to sign this request? yes

   Successfully wrote the signed certificate to file
   '/ca/intermediate-ca-cert.pem'.
   ```

5. After you obtain the intermediate CA certificate, create secure, offline backups of the root CA certificate.

6. Remove the root CA certificate, or at least its private key, from all systems.

> ⓘ **Note:**
>
> Make certain that all end-entity certificates are signed with the intermediate CA certificate, and that the process is identical to the previous example. Restore the root CA certificate only if you need to sign another intermediate CA certificate.

Enabling TLS support during setup
Enable TLS support in the server.

To enable TLS support in the server, you should complete one of the following tasks during the setup procedure:

- Provide a key store that contains the certificate to use.
- Make the installer generate a self-signed certificate.

When using the **`setup`** tool in interactive mode, it prompts you for the information that it needs to configure secure communication, as shown in the following example.

```
Do you want to enable the Directory Server services (Available State,
Available or Degraded State, Configuration, Consent, Directory REST API,
Documentation, SCIM2, and Swagger UI) and Administrative Console over
HTTPS?  After setup, you can selectively enable or disable individual
services and applications by configuring the HTTPS Connection Handler
(yes / no) [yes]: yes

On which port should the Directory Server accept connections from HTTPS
clients? [443]: 443

On which port should the Directory Server accept connections from LDAP
clients? [389]: 389

Do you want to enable LDAPS? (yes / no) [yes]: yes
On which port should the Directory Server accept connections from LDAPS
clients? [636]: 636

Do you want to enable StartTLS? (yes / no) [yes]: yes

Certificate server options:

    1)  Generate self-signed certificate (recommended for testing purposes
        only)
    2)  Use an existing certificate located on a Java Keystore (JKS)
    3)  Use an existing certificate located on a PKCS12 keystore
    4)  Use an existing certificate on a PKCS11 token

Enter option [1]: 2

Java Keystore (JKS) path: /ca/ds1-keystore
Keystore PIN: {password}

Truststore options:

    1)  Generate a default JKS truststore
    2)  Use an existing JKS truststore
    3)  Use an existing PKCS12 truststore

Enter option [1]: 2

JKS truststore path: /ca/truststore
```

```
Truststore password (can be blank): {password}
```

When using **setup** in non-interactive mode, use the following arguments to configure TLS support.

Argument	Description
`--ldapsPort {port}`	Server enables support for LDAPS (LDAP over TLS) on the specified TCP port.
`--httpsPort {port}`	Server enables support for HTTPS for SCIM, the Directory REST API, and the web-based administration console on the specified TCP port.
`--enableStartTLS`	LDAP connection handler enables support for the StartTLS extended operation.
`--generateSelfSignedCertificate`	**setup** generates a self-signed certificate that is presented to clients that use LDAPS, HTTPS, and the StartTLS extended operation.
`--useJavaKeyStore {path}`	Server uses the specified Java KeyStore (JKS) key store to obtain the certificate chain that it presents to clients that use LDAPS, HTTPS, and the StartTLS extended operation.
`--usePKCS12KeyStore {path}`	Server uses the specified PKCS #12 key store to obtain the certificate chain that it presents to clients that use LDAPS, HTTPS, and the StartTLS extended operation.
`--usePKCS11KeyStore`	Server uses a PKCS #11 key store, like a hardware security module, to obtain the certificate chain that it presents to clients that use LDAPS, HTTPS, and the StartTLS extended operation. The Java Virtual Machine (JVM) must already be configured to access the appropriate key store through PKCS #11.
`--keyStorePassword {password}`	Password that is needed to interact with the specified JKS, PKCS #12, or PKCS #11 key store. The **setup** tool assumes that the private key password matches the key store password.
`--keyStorePasswordFile {path}`	Path to the file that contains the password needed to interact with the specified JKS, PKCS #12, or PKCS #11 key store.
`--certNickname {alias}`	Alias of the private key entry in the specified key store that contains the certificate chain to present to clients during TLS negotiation. This argument is optional but recommended if the key store contains multiple certificates.
`--useJavaTrustStore {path}`	Server uses the specified JKS trust store to determine whether to trust certificate chains that are presented to it during TLS negotiation.

Argument	Description
`--usePKCS12TrustStore {path}`	Server uses the specified PKCS #12 trust store to determine whether to trust certificate chains that are presented to it during TLS negotiation
`--trustStorePassword {password}`	Password that is needed to interact with the specified JKS or PKCS #11 trust store.
`--trustStorePasswordFile {path}`	Path to the file that contains the password needed to interact with the specified JKS or PKCS #11 trust store.

The following example command sets up PingDirectory Server in non-interactive mode with an existing certificate.

```
$ ./setup \
     --no-prompt \
     --acceptLicense \
     --ldapPort 389 \
     --ldapsPort 636 \
     --httpsPort 443 \
     --enableStartTLS \
     --useJavaKeyStore config/keystore \
     --keyStorePasswordFile config/keystore.pin \
     --certNickname server-cert \
     --useJavaTrustStore config/truststore \
     --trustStorePasswordFile config/truststore.pin \
     --baseDN dc=example,dc=com \
     --rootUserDN "cn=Directory Manager" \
     --rootUserPasswordFile root-pw.txt \
     --maxHeapSize 10g \
     --encryptDataWithPassphraseFromFile encryption-settings-password.txt \
     --instanceName ds1 \
     --location Austin \
     --noPropertiesFile


Ping Identity Directory Server 8.0.0.0

Initializing ..... Done
Configuring Directory Server ..... Done
Configuring Certificates ..... Done
Starting Directory Server ..... Done

Access product documentation from docs/index.html
```

Enabling TLS support after setup

If the server has been set up without support for TLS, enable TLS support later by completing the following tasks.

Steps

1. Obtain a certificate chain.

   For more information about obtaining a certificate chain, see *Certificate chains* on page 322. To prepare a Java KeyStore JKS or PKCS #12 key store with an appropriate certificate chain and private key, use the **`manage-certificates`** tool. We also recommend that you create a trust store that the server can use.

2. Configure the key and trust manager providers.

   For more information, see *Configuring key and trust manager providers* on page 350.
3. Configure connection handlers.

   For more information, see *Configuring connection handlers* on page 350.

Configuring key and trust manager providers
After you have a key store, configure a key manager provider to access it.

The server is preconfigured with key manager providers, `JKS` and `PKCS12`, that you can use with JKS or PKCS #12 key stores, respectively. You can update the appropriate key manager provider in most cases to reference the key store that you plan to use. The following code provides an example.

```
dsconfig set-key-manager-provider-prop \
     --provider-name JKS \
     --set enabled:true \
     --set key-store-file:config/keystore \
     --set key-store-pin-file:config/keystore.pin
```

A similar change configures a trust manager provider to reference the appropriate trust store. The following code provides an example.

```
dsconfig set-trust-manager-provider-prop \
     --provider-name JKS \
     --set enabled:true \
     --set include-jvm-default-issuers:true \
     --set trust-store-file:config/truststore
     --set trust-store-pin-file:config/truststore.pin
```

ⓘ **Note:**

If all clients and servers use certificates that are signed by issuers and are included in the JVM's default trust store, you can use the `JVM-Default` trust manager provider to accomplish this task.

Configuring connection handlers
After you configure the key and trust manager providers, update the connection handlers to use the key and trust manager providers.

Steps

- For the LDAP connection handler, use the following command to enable StartTLS with a configuration change. By default, the LDAP connection handler accepts non-secure connections.

  ```
  dsconfig set-connection-handler-prop \
    --handler-name "LDAP Connection Handler" \
    --set allow-start-tls:true \
    --set key-manager-provider:JKS \
    --set trust-manager-provider:JKS \
    --set ssl-cert-nickname:server-cert \
    --set ssl-client-auth-policy:optional
  ```
- If you did not configure secure communication during setup, the LDAPS connection handler is disabled. To configure LDAPS support in this scenario, enable the connection handler and configure most of the same settings. You must set allow-start-tls to `false` and use-ssl to `true`. See the following code for an example configuration.

  ```
  dsconfig set-connection-handler-prop \
    --handler-name "LDAPS Connection Handler" \
    --set enabled:true \
  ```

```
  --set key-manager-provider:JKS \
  --set trust-manager-provider:JKS \
  --set ssl-cert-nickname:server-cert \
  --set ssl-client-auth-policy:optional
```

The following example uses a similar configuration change to enable the HTTPS connection handler.

```
dsconfig set-connection-handler-prop \
  --handler-name "HTTPS Connection Handler" \
  --set enabled:true \
  --set listen-port:443 \
  --set key-manager-provider:JKS \
  --set trust-manager-provider:JKS \
  --set ssl-cert-nickname:server-cert
```

Updating the topology registry

After the server connection handlers are updated to enable TLS, update the topology registry to provide information about the new configuration.

The topology registry holds information about server instances that are part of the environment, and it helps to facilitate inter-server communication, such as replication, mirroring portions of the configuration, and PingDirectory Server's automatic backend server-discovery functionality.

The following table details the two types of entries that require updating.

Configuration types and their update descriptions

Configuration Type	Update description

Server instance listener configuration	▪ Provides information that is needed to trust the TLS certificates that instances in the topology present.
	▪ The server instance listener configuration must include the server certificate, which is defined as the certificate at the head of the chain. This version must be the multi-line, PEM-formatted representation of the certificate. You can use **`dsconfig`** to import the certificate from a file, as shown in the following example.
	`bin/dsconfig set-server-instance-listener-prop \`
	`  --instance-name ds1 \`
	`  --listener-name ldap-listener-mirrored-config \`
	`  --set server-ldap-port:636 \`
	`  --set connection-security:ssl \`
	`  --set 'listener-certificate>/ca/ds1-cert.pem'`
	Note:
	The less-than operator > in the final line indicates that the value is read from a file rather than provided directly. In addition, you might not need to enclose the property name and path within single straight quotes to prevent the shell from interpreting the less-than symbol as an attempt to redirect input.

Server instance configuration	▪ Provides information about options for communicating with those instances.
	▪ Update the server instance configuration object to reflect the new methods that are available for communication with the instance. For example, the `preferred-security` property identifies the mechanism by which other instances in the topology attempt to communicate with the instance.
	The following example code sets the LDAPS and HTTPS ports, indicates that StartTLS support is enabled, and instructs other instances to use SSL (LDAPS) when communicating with the instance.
	`dsconfig set-server-instance-prop \`
	`  --instance-name ds1 \`
	`  --set ldaps-port:636 \`
	`  --set https-port:443 \`
	`  --set preferred-security:ssl \`
	`  --set start-tls-enabled:true`

Troubleshooting TLS-related issues

Use this section for troubleshooting problems that might arise during TLS configuration, including communication and security issues that affect clients as well as PingDirectory Server.

- *Log messages*
- *manage-certificates check-certificate-usability*
- *ldapsearch*
- *Using low-level TLS debugging*

Log messages

The following describes how to use the server's log messages to troubleshoot TLS-related issues.

To troubleshoot TLS-related issues, start by checking the server's access log. If the client can establish a TCP connection to the server, which must occur before TLS negotiation can start, the access log shows a `CONNECT` message with the following information:

- Source and destination address and port for the connection
- Protocol
- Selected client connection policy

The `CONNECT` message does not appear

If the `CONNECT` does not appear, the client might be unable to communicate with the server. The culprit can be a network problem, a firewall that is blocking attempts to communicate, or the client is trying to use an incorrect address or port.

The `CONNECT` message does appear

If the `CONNECT` message appears in the access log, it typically includes a `conn` element that specifies the connection ID. To view additional log messages for the client connection, use the **`search-logs`** tool. For example, if the connection ID is `12345`, the following command displays the complete set of associated log messages.

```
$ bin/search-logs --logFile logs/access conn=12345
```

If you are using LDAPS

If you are attempting to use LDAPS, one of the following log messages appears next:

- `SECURITY-NEGOTIATION` message – Indicates that the client and server successfully completed the negotiation process and that the issue likely occurred after the TLS session was established. This message also includes details about the negotiation, including the TLS protocol and the selected cipher suite.
- `DISCONNECT` message – The issue might involve a failure in the TLS-negotiation process. In such scenarios, the message usually includes a `reason` element that provides additional information about the reason for the disconnect.

  If the failure occurred during TLS negotiation, the usefulness of the `DISCONNECT` message depends in part on whether the failure occurred on the client or the server. For example, if the server decided to abort the negotiation, the message ideally contains the specific reason. If the problem occurred on the client, the log message likely contains only the general category for the failure.

> ⓘ **Note:**
>
> The TLS protocol does not provide a mechanism for conveying detailed error messages. Instead, it offers only a basic alert mechanism with a fixed set of alert types. For example, if a client does not trust the certificate chain that the server presents to it, the server might receive a generic alert like `certificate_unknown`, even if the client knows the precise reason for rejecting the chain. In such

instances, you might need to determine whether the client can provide additional details about the issue.

If the access log does not provide useful information

If the access log does not provide useful information, check the server error log. Although the error log does not normally include information about issues that relate to client communication, it provides helpful information in certain circumstances, like when an internal error within the server interferes with communication attempts.

manage-certificates check-certificate-usability
The **`manage-certificates`** tool offers a **`check-certificate-usability`** subcommand to examine a specified entry in a key store and to identify potential issues that might interfere with secure communication.

The **`check-certificate-usability`** tool completes the following tasks:

- Ensures that a specified entry in the key store includes a private key and a complete certificate chain
- Checks whether the certificate at the root of the chain is found in the Java virtual machine's (JVM's) default set of trusted certificates
- Ensures that the current time lies is within the validity window for all certificates in the chain
- Validates the signatures for all certificates in the chain
- Warns if the end-entity certificate is self-signed
- Warns if the end-entity certificate does not contain an extended key usage extension with the `serverAuth` usage
- Warns if the issuer certificates do not have a key usage extension with the `keyCertSign` usage
- Warns if the issuer certificates do not have a basic constraints extension indicating that it can operate as a certification authority

  If the chain violates a path length constraint, the **`check-certificate-usability`** tool reports an error.
- Ensures that the signature algorithm uses a strong message digest algorithm, like SHA-256

  The **`check-certificate-usability`** tool reports an error for weak digest algorithms like MD5 or SHA-1, and reports a warning for unrecognized digest algorithms.
- Ensures that none of the certificates that use an RSA key pair have a key size less than 2048 bits

The following example demonstrates the usage for the **`manage-certificates check-certificate-usability`** command and its output when no problems are identified.

```
$ bin/manage-certificates check-certificate-usability \
     --keystore config/keystore \
     --keystore-password-file config/keystore.pin \
     --alias server-cert

Successfully retrieved the certificate chain for alias 'server-cert':

Subject DN:  CN=ds1.example.com,O=Example Corp,C=US
Issuer DN:  CN=Example Intermediate CA,O=Example Corp,C=US
Validity Start Time: Tuesday, November 12, 2019 at 03:52:44 PM CST
                     (5 minutes, 45 seconds ago)
Validity End Time: Wednesday, November 11, 2020 at 03:52:44 PM CST
                   (364 days, 23 hours, 54 minutes, 14 seconds from now)
Validity State:  The certificate is currently within the validity window.
Signature Algorithm:  SHA-256 with RSA
Public Key Algorithm:  RSA (2048-bit)
SHA-1 Fingerprint:
 84:e4:00:b9:f0:6b:58:bb:ac:67:79:28:2f:43:9f:e3:ac:24:ee:98
SHA-256 Fingerprint:
 63:85:4d:2c:50:ea:a8:84:54:e0:73:9a:e7:5b:e7:1b:06:85:0e:
```

```
                    28:2b:76:a9:8b:57:fc:27:f7:60:81:48:41

Subject DN:  CN=Example Intermediate CA,O=Example Corp,C=US
Issuer DN:  CN=Example Root CA,O=Example Corp,C=US
Validity Start Time: Tuesday, November 12, 2019 at 03:52:42 PM CST
                     (5 minutes, 47 seconds ago)
Validity End Time: Monday, November 7, 2039 at 03:52:42 PM CST
                   (7299 days, 23 hours, 54 minutes, 12 seconds from now)
Validity State:  The certificate is currently within the validity window.
Signature Algorithm:  SHA-256 with RSA
Public Key Algorithm:  RSA (4096-bit)
SHA-1 Fingerprint:
 de:da:3d:fc:d4:1f:67:79:0a:a1:5a:cd:ca:4a:7e:a5:d3:46:88:27
SHA-256 Fingerprint:

 02:3c:af:ad:b7:07:81:89:45:48:d0:09:31:a8:90:c4:17:11:1c:00:11:fd:49:b2:2c:
   ba:ac:dd:c4:9f:03:36

Subject DN:  CN=Example Root CA,O=Example Corp,C=US
Issuer DN:  CN=Example Root CA,O=Example Corp,C=US
Validity Start Time: Tuesday, November 12, 2019 at 03:52:38 PM CST
                     (5 minutes, 51 seconds ago)
Validity End Time: Monday, November 7, 2039 at 03:52:38 PM CST
                   (7299 days, 23 hours, 54 minutes, 8 seconds from now)
Validity State:  The certificate is currently within the validity window.
Signature Algorithm:  SHA-256 with RSA
Public Key Algorithm:  RSA (4096-bit)
SHA-1 Fingerprint:
 8e:03:e4:58:e6:e3:59:9a:55:77:c0:88:3c:fa:d7:29:f4:ff:de:6c
SHA-256 Fingerprint:
 95:54:0d:e2:aa:48:29:c1:25:7c:20:69:c0:27:33:31:81:07:02:
                    2e:00:24:ae:49:5e:98:bd:a3:72:a5:05:26

OK:  The certificate chain is complete.  Each subsequent certificate is
the issuer for the previous certificate in the chain, and the chain ends
with a self-signed certificate.

OK:  Certificate 'CN=ds1.example.com,O=Example Corp,C=US' has a valid
signature.

OK:  Certificate 'CN=Example Intermediate CA,O=Example Corp,C=US' has a
valid signature.

OK:  Certificate 'CN=Example Root CA,O=Example Corp,C=US' has a valid
signature.

OK:  Certificate 'CN=ds1.example.com,O=Example Corp,C=US' will expire at
Wednesday, November 11, 2020 at 03:52:44 PM CST (364 days, 23 hours, 54
minutes, 14 seconds from now), which is not in the near future.

OK:  Issuer certificate 'CN=Example Intermediate CA,O=Example Corp,C=US'
will expire at Monday, November 7, 2039 at 03:52:42 PM CST (7299 days, 23
hours, 54 minutes, 12 seconds from now), which is not in the near future.

OK:  Issuer certificate 'CN=Example Root CA,O=Example Corp,C=US' will
expire at Monday, November 7, 2039 at 03:52:38 PM CST (7299 days, 23
hours, 54 minutes, 8 seconds from now), which is not in the near future.

OK:  Certificate 'CN=ds1.example.com,O=Example Corp,C=US' at the head of
the chain includes an extended key usage extension, and that extension
includes the serverAuth usage.

OK:  Issuer certificate 'CN=Example Intermediate CA,O=Example Corp,C=US'
includes a basic constraints extension, and the certificate chain
```

```
satisfies those constraints.

OK:  Issuer certificate 'CN=Example Intermediate CA,O=Example Corp,C=US'
includes a key usage extension with the keyCertSign usage flag set to
true.

OK:  Issuer certificate 'CN=Example Root CA,O=Example Corp,C=US' includes
a basic constraints extension, and the certificate chain satisfies those
constraints.

OK:  Issuer certificate 'CN=Example Root CA,O=Example Corp,C=US' includes
a key usage extension with the keyCertSign usage flag set to true.

OK:  Certificate 'CN=ds1.example.com,O=Example Corp,C=US' uses a signature
algorithm of 'SHA-256 with RSA', which is is considered strong.

OK:  Certificate 'CN=Example Intermediate CA,O=Example Corp,C=US' uses a
signature algorithm of 'SHA-256 with RSA', which is is considered strong.

OK:  Certificate 'CN=Example Root CA,O=Example Corp,C=US' uses a signature
algorithm of 'SHA-256 with RSA', which is is considered strong.

OK:  Certificate 'CN=ds1.example.com,O=Example Corp,C=US' has a 2048-bit
RSA public key, which is considered strong.

OK:  Certificate 'CN=Example Intermediate CA,O=Example Corp,C=US' has a
4096-bit RSA public key, which is considered strong.

OK:  Certificate 'CN=Example Root CA,O=Example Corp,C=US' has a 4096-bit
RSA public key, which is considered strong.

No usability errors or warnings were identified while validating the
certificate chain.
```

If any usability issues are identified, they might be responsible for communication problems.

ldapsearch

The **`ldapsearch`** command-line utility is a powerful tool for issuing searches against an LDAP directory server. It also provides a convenient method for troubleshooting a variety of issues, including problems that are relevant to TLS communication.

The following table details arguments that are the most useful for TLS-related communication.

TLS-related communication arguments and their descriptions

Argument	Description
--hostname {address}	Address of the server to which the connection is established
--port {port}	TCP port of the server to which the connection is established. The standard port for non-secure LDAP, or LDAP to be secured with StartTLS, is 389, and the standard port for secure LDAPS is 636. Many deployments use alternate ports, especially non-privileged ports above 1024.
--useSSL	The tool establishes an initially insecure LDAP connection, which is secured later with the StartTLS extended operation.

Argument	Description
--trustStorePath {path}	Path to the trust store that is used when determining whether to trust the certificate chain that the server presents during TLS negotiation. If neither this argument nor the --trustAll argument is provided, the tool prompts you interactively whether to trust server certificates that are not signed by an issuer in the Java virtual machine's (JVM's) default trust store.
--trustStoreFormat {format}	Format for the trust store, which is typically JKS or PKCS12.
--trustStorePassword {password}	Password that is required to access the contents of the trust store.
--trustStorePasswordFile {path}	Path to the file that contains the password that is required to access the contents of the trust store.
--trustAll	The tool blindly trusts all TLS certificate chains that are presented to it. Although this argument can prove useful for troubleshooting purposes, it is not recommended for general use.
--keyStorePath {path}	Path to the key store to use if a client certificate chain is presented to the server. ⓘ **Note:** Use this argument only when one of the following conditions is satisfied: ▪ The server is configured to require clients to present a certificate. ▪ You intend to use the certificate to authenticate through SASL EXTERNAL.
--keyStoreFormat {format}	Format for the key store, which is typically JKS or PKCS12.
--keyStorePassword {password}	Password to access the key store.
--keyStorePasswordFile {path}	Path to the file that contains the password necessary to access the key store.
--certNickname {alias}	Alias of the private key entry in the key store. Use when obtaining the certificate chain to present to the server.
--useSASLExternal	The client authenticates with the EXTERNAL SASL mechanism, which typically identifies the client using the certificate chain that is presented during TLS negotiation.

Argument	Description
--enableSSLDebugging	The tool activates the low-level TLS-debugging feature that the JVM provides.

The following command provides an example of the simplest method for testing TLS communication with PingDirectory Server.

```
$ bin/ldapsearch \
     --hostname ds1.example.com \
     --port 636 \
     --useSSL \
     --baseDN "" \
     --scope base \
     "(objectClass=*)"
The server presented the following certificate chain:

     Subject: CN=ds1.example.com,O=Example Corp,C=US
     Valid From: Tuesday, November 12, 2019 at 08:28:08 PM CST
     Valid Until: Wednesday, November 11, 2020 at 08:28:08 PM
 CST
     SHA-1 Fingerprint:

 6a:22:2a:bd:0b:1b:09:35:63:bc:12:3e:2c:9e:e7:70:bc:a4:73:de
     256-bit SHA-2 Fingerprint:

 7a:8c:e4:76:d4:47:15:fd:65:f5:26:0e:d2:55:77:d7:03:7a:e6:79:9f:bc:
        ae:93:2c:76:9c:01:fc:ef:15:38
     -
     Issuer 1 Subject: CN=Example Intermediate CA,O=Example
 Corp,C=US
     Valid From: Tuesday, November 12, 2019 at 08:28:06 PM CST
     Valid Until: Monday, November 7, 2039 at 08:28:06 PM CST
     SHA-1 Fingerprint:
 01:b3:70:8b:6c:11:43:87:3b:e9:bb:73:27:99:ea:fd:08:c4:db:ec
     256-bit SHA-2 Fingerprint:
 49:60:69:df:33:9d:26:d0:66:c9:6d:7b:0b:cb:3b:96:

 40:22:dc:6d:11:32:b7:c0:30:47:d6:7c:6a:19:cd:60
     -
     Issuer 2 Subject: CN=Example Root CA,O=Example Corp,C=US
     Valid From: Tuesday, November 12, 2019 at 08:28:03 PM CST
     Valid Until: Monday, November 7, 2039 at 08:28:03 PM CST
     SHA-1 Fingerprint:
 b4:83:55:db:82:e4:63:5c:3a:44:13:8f:88:44:e3:60:f2:53:80:48
     256-bit SHA-2 Fingerprint:

 e8:af:6f:ed:b9:0e:df:94:9c:20:29:53:a9:74:44:a9:17:b4:08:65:c8:19:c1:fb:
        34:34:a1:90:83:8a:d5:12

Do you wish to trust this certificate?  Enter 'y' or 'n': y
dn:
objectClass: top
objectClass: ds-root-dse
startupUUID: 8d574122-4584-4522-96d9-0cdcb9d2e339
startTime: 201911130611492

# Result Code:  0 (success)
# Number of Entries Returned:  1
```

Trust stores and trust-related arguments

If no trust-related arguments are provided, the tool uses the JVM's default trust store to verify whether to trust the certificate chain, based on the information that it contains. If a trusted authority has signed the server certificate, the negotiation process continues without further interaction.

If the chain cannot be trusted, based on the information in the JVM-default trust store, **ldapsearch** prompts you interactively about whether to trust the certificate. If you accept the chain, the client and server complete the negotiation process, and the client sends the search request to the server. If the search succeeds, the server can communicate over TLS.

To test with a trust store instead of being prompted interactively, use the --trustStorePath argument that points to the appropriate trust store. If you are using a Java Keystore (JKS) trust store, you might not need to provide the trust store password. If you are using a PKCS #12 trust store, you need to provide the trust store password. The following code provides an example.

```
$ bin/ldapsearch \
     --hostname ds1.example.com \
     --port 636 \
     --useSSL \
     --trustStorePath config/truststore.p12 \
     --trustStorePasswordFile config/truststore.pin \
     --trustStoreFormat PKCS12 \
     --baseDN "" \
     --scope base \
     "(objectClass=*)"
dn:
objectClass: top
objectClass: ds-root-dse
startupUUID: c8724159-8c37-45eb-b210-879bfcf74ad6
startTime: 20191113154023Z

# Result Code:  0 (success)
# Number of Entries Returned:  1
```

Client certificate chains and key stores

To present a client certificate chain to the server, either because the server's connection handler is configured with an ssl-client-auth-policy value of required or because you plan to use the certificate to authenticate by way of the SASL EXTERNAL mechanism, provide at least the key store and its corresponding password. You can also specify the alias of the certificate chain to present, which is recommended if your client key store contains multiple certificates. The following code provides an example.

```
$ bin/ldapsearch \
     --hostname ds1.example.com \
     --port 636 \
     --useSSL \
     --trustStorePath config/truststore.p12 \
     --trustStorePasswordFile config/truststore.pin \
     --trustStoreFormat PKCS12 \
     --keyStorePath client-keystore \
     --keyStorePasswordFile client-keystore.pin \
     --certNickname client-cert \
     --useSASLExternal \
     --baseDN "" \
     --scope base \
     "(objectClass=*)"
```

```
dn:
objectClass: top
objectClass: ds-root-dse
startupUUID: c8724159-8c37-45eb-b210-879bfcf74ad6
startTime: 20191113154023Z

# Result Code:  0 (success)
# Number of Entries Returned:  1
```

If you need to further troubleshoot a TLS-related issue

If you encounter a TLS-related issue that you cannot resolve by examining the **`ldapsearch`** output or the server logs, use the --enableSSLDebugging option to enable the JVM's support for low-level debugging of TLS processing. For more information, see *Using low-level TLS debugging* on page 360.

Using low-level TLS debugging
Use tools other than the command-line tools that are provided with PingDirectory Server for performing low-level TLS debugging.

Before you begin

> ⓘ **Note:**
>
> If you need to use low-level debugging options, enable the Java Virtual Machine (JVM)'s support for TLS debugging. Many of the command-line tools that are provided with PingDirectory Server, such as **`ldapsearch`**, offer an --enableSSLDebugging argument that simplifies this process.

Steps

1. In the `config/java.properties` file, add the following line to the set of properties for the appropriate tool.

   ```
   -Djavax.net.debug=all
   ```

2. For the changes to take effect, run the **`bin/dsjavaproperties`** command.

Next steps

The next time the tool is run, an output is generated detailing the TLS-related processing that the JVM is performing. You and the support team can use the output to identify the issue.

Using the Configuration API

PingDirectory Server provides a Configuration API when updating the server configuration with LDAP is not possible. The API is consistent with the System for Cross-domain Identity Management (SCIM) 2.0 protocol and uses JSON as a text exchange format, so all request headers allow the application/json content type.

About this task
The server includes a servlet extension that provides read and write access to the server's configuration over HTTP.

Steps

- To add the extension to one of the server's HTTP Connection Handlers, run the following code.

  ```
  $ bin/dsconfig set-connection-handler-prop \
  ```

```
  --handler-name "HTTPS Connection Handler" \
  --add http-servlet-extension:Configuration
```

> ⓘ **Note:**
>
> By default, the extension is enabled for new installations. You can enable the extension for existing deployments.

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

Authentication and authorization with the Configuration API
Use this topic for how to make changes for customizing authentication and authorization access with the Configuration API.

Authentication

Clients must use HTTP basic authentication to authenticate to the Configuration API. If the username value is not a distinguished name (DN), then it resolves to a DN value using the identity mapper associated with the Configuration servlet. By default, the Configuration API uses an identity mapper that allows an entry's UID value to be used as a username. To customize this behavior, either customize the default identity mapper or specify a different identity mapper using the Configuration servlet's `identity-mapper` property. The following code provides an example.

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

Authorization

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

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

The Configuration API and the dsconfig tool relationship
The Configuration API is designed to mirror the **`dsconfig`** tool, using the same names for properties and object types.

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

The API includes the filter, sort, and pagination query parameters described by the System for Cross-domain Identity Management (SCIM) specification. Request specific attributes using the attributes query parameter, whose value must be a comma-delimited list of properties to be returned, such as `attributes=baseDN,description`. You can exclude attributes from responses by specifying the excludedAttributes parameter.

Configuration API operations supported in REST APIs

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

ⓘ **Note:**

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

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

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

GET example

This topic provides an example of a GET request and response concerning the userRoot backend for reference.

GET request

The following code example is a GET request for information about the **`userRoot`** backend.

```
GET /config/backends/userRoot
```

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

GET response

The following code example is the response.

```
{
  "schemas": [
    "urn:unboundid:schemas:configuration:2.0:backend:local-db"
  ],
   "id": "userRoot",
   "meta": {
     "resourceType": "Local DB Backend",
     "location": "http://localhost:5033/config/backends/userRoot"
   },
   "backendID": "userRoot2",
   "backgroundPrime": "false",
   "backupFilePermissions": "700",
   "baseDN": [
     "dc=example2,dc=com"
   ],
   "checkpointOnCloseCount": "2",
   "cleanerThreadWaitTime": "120000",
   "compressEntries": "false",
   "continuePrimeAfterCacheFull": "false",
   "dbBackgroundSyncInterval": "1 s",
   "dbCachePercent": "10",
   "dbCacheSize": "0 b",
   "dbCheckpointerBytesInterval": "20 mb",
   "dbCheckpointerHighPriority": "false",
   "dbCheckpointerWakeupInterval": "1 m",
   "dbCleanOnExplicitGC": "false",
   "dbCleanerMinUtilization": "75",
   "dbCompactKeyPrefixes": "true",
   "dbDirectory": "db",
   "dbDirectoryPermissions": "700",
   "dbEvictorCriticalPercentage": "0",
   "dbEvictorLruOnly": "false",
   "dbEvictorNodesPerScan": "10",
   "dbFileCacheSize": "1000",
   "dbImportCachePercent": "60",
   "dbLogFileMax": "50 mb",
   "dbLoggingFileHandlerOn": "true",
   "dbLoggingLevel": "CONFIG",
   "dbNumCleanerThreads": "0",
   "dbNumLockTables": "0",
   "dbRunCleaner": "true",
   "dbTxnNoSync": "false",
   "dbTxnWriteNoSync": "true",
   "dbUseThreadLocalHandles": "true",
   "deadlockRetryLimit": "10",
   "defaultCacheMode": "cache-keys-and-values",
   "defaultTxnMaxLockTimeout": "10 s",
   "defaultTxnMinLockTimeout": "10 s",
   "enabled": "false",
   "explodedIndexEntryThreshold": "4000",
   "exportThreadCount": "0",
   "externalTxnDefaultBackendLockBehavior": "acquire-before-retries",
   "externalTxnDefaultMaxLockTimeout": "100 ms",
   "externalTxnDefaultMinLockTimeout": "100 ms",
   "externalTxnDefaultRetryAttempts": "2",
   "hashEntries": "false",
```

```
    "id2childrenIndexEntryLimit": "66",
    "importTempDirectory": "import-tmp",
    "importThreadCount": "16",
    "indexEntryLimit": "4000",
    "isPrivateBackend": "false",
    "javaClass": "com.unboundid.directory.server.backends.jeb.BackendImpl",
    "jeProperty": [
      "je.cleaner.adjustUtilization=false",
      "je.nodeMaxEntries=32"
    ],
    "numRecentChanges": "50000",
    "offlineProcessDatabaseOpenTimeout": "1 h",
    "primeAllIndexes": "true",
    "primeMethod": [
      "none"
    ],
    "primeThreadCount": "2",
    "primeTimeLimit": "0 ms",
    "processFiltersWithUndefinedAttributeTypes": "false",
    "returnUnavailableForUntrustedIndex": "true",
    "returnUnavailableWhenDisabled": "true",
    "setDegradedAlertForUntrustedIndex": "true",
    "setDegradedAlertWhenDisabled": "true",
    "subtreeDeleteBatchSize": "5000",
    "subtreeDeleteSizeLimit": "5000",
    "uncachedId2entryCacheMode": "cache-keys-only",
    "writabilityMode": "enabled"
}
```


GET list example

See the following example of a GET request and response for all local backends for reference.

GET request

The following is a code example GET request for all local backends.

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

GET response

The following is a code example GET response, which is shortened.


```
{
  "schemas": [
    "urn:ietf:params:scim:api:messages:2.0:ListResponse"
  ],
  "totalResults": 24,
  "Resources": [
    {
      "schemas": [
        "urn:unboundid:schemas:configuration:2.0:backend:ldif"
      ],
      "id": "adminRoot",
      "meta": {
        "resourceType": "LDIF Backend",
        "location": "http://localhost:5033/config/backends/adminRoot"
      },
      "backendID": "adminRoot",
      "backupFilePermissions": "700",
      "baseDN": [
        "cn=topology,cn=config"
```

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


PATCH example

Modify the Configuration API using the HTTP PATCH method.

The PATCH request body is a JSON object formatted according to the System for Cross-domain Identity Management (SCIM) patch request. The Configuration API supports a subset of possible values for the path attribute that indicates the configuration attribute to modify.

Modify the configuration object's attributes according to the information in the following table, which details the comparable `dsconfig modify-[object]` options to each PATCH request.

Operations and PATCH requests to modify the configuration object's attributes

Operation	PATCH request	Comparable `dsconfig modify-[object]` options
Set the single-valued `description` attribute to a new value.	`{ "op" : "replace", "path" : "description", "value" : "A new backend." }`	`$ dsconfig set-backend-prop --backend-name userRoot \ --set "description:A new backend"`
Add a new value to the multi-valued `jeProperty` attribute.	`{ "op" : "add", "path" : "jeProperty", "value" : "je.env.backgroundReadLi`	`$ dsconfig set-backend-prop --backend-name userRoot \ --add je-property:je.env.backgroundReadLimi`
Remove a value from a multi-valued property. In this case, path specifies a SCIM filter identifying the value to remove.	`{ "op" : "remove", "path" : "[jeProperty eq \"je.cleaner.adjustUtili \"]" }`	`$ dsconfig set-backend-prop --backend-name userRoot \ --remove je-property:je.cleaner.adjustUtilizat`
Second operation to remove a value from a multi-valued property, where the `path` specifies both an attribute to modify and a SCIM filter whose attribute is the following value.	`{ "op" : "remove", "path" : "jeProperty[value eq \"je.nodeMaxEntries=32\"]`	`$ dsconfig set-backend-prop --backend-name userRoot \ --remove je-property:je.nodeMaxEntries=32`
Option to remove one or more values of a multi-valued attribute. This has the effect of restoring the attribute's value to its default value.	`{ "op" : "remove", "path" : "id2childrenIndexEntryLi }`	`$ dsconfig set-backend-prop --backend-name userRoot \ --reset id2childrenIndexEntryLimit`

The following is the full example request.

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

{
  "schemas" :
 [ "urn:ietf:params:scim:api:messages:2.0:PatchOp" ],
  "Operations" : [ {
    "op" : "replace",
    "path" : "description",
    "value" : "A new backend."
```

```
  }, {
    "op" : "add",
    "path" : "jeProperty",
    "value" : "je.env.backgroundReadLimit=0"
  }, {
    "op" : "remove",
    "path" : "[jeProperty eq
 \"je.cleaner.adjustUtilization=false\"]"
  }, {
    "op" : "remove",
    "path" : "jeProperty[value eq \"je.nodeMaxEntries=32\"]"
  }, {
    "op" : "remove",
    "path" : "id2childrenIndexEntryLimit"
  } ]
}
```

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

```
{
 "schemas": [
   "urn:unboundid:schemas:configuration:2.0:backend:local-db"
 ],
 "id": "userRoot2",
 "meta": {
   "resourceType": "Local DB Backend",
   "location": "http://example.com:5033/config/backends/
userRoot2"
 },
 "backendID": "userRoot2",
 "backgroundPrime": "false",
 "backupFilePermissions": "700",
 "baseDN": [
   "dc=example2,dc=com"
 ],
 "checkpointOnCloseCount": "2",
 "cleanerThreadWaitTime": "120000",
 "compressEntries": "false",
 "continuePrimeAfterCacheFull": "false",
 "dbBackgroundSyncInterval": "1 s",
 "dbCachePercent": "10",
 "dbCacheSize": "0 b",
 "dbCheckpointerBytesInterval": "20 mb",
 "dbCheckpointerHighPriority": "false",
 "dbCheckpointerWakeupInterval": "1 m",
 "dbCleanOnExplicitGC": "false",
 "dbCleanerMinUtilization": "75",
 "dbCompactKeyPrefixes": "true",
 "dbDirectory": "db",
 "dbDirectoryPermissions": "700",
 "dbEvictorCriticalPercentage": "0",
 "dbEvictorLruOnly": "false",
 "dbEvictorNodesPerScan": "10",
 "dbFileCacheSize": "1000",
 "dbImportCachePercent": "60",
 "dbLogFileMax": "50 mb",
 "dbLoggingFileHandlerOn": "true",
 "dbLoggingLevel": "CONFIG",
 "dbNumCleanerThreads": "0",
```

```
 "dbNumLockTables": "0",
 "dbRunCleaner": "true",
 "dbTxnNoSync": "false",
 "dbTxnWriteNoSync": "true",
 "dbUseThreadLocalHandles": "true",
 "deadlockRetryLimit": "10",
 "defaultCacheMode": "cache-keys-and-values",
 "defaultTxnMaxLockTimeout": "10 s",
 "defaultTxnMinLockTimeout": "10 s",
 "description": "123",  "enabled": "false",
 "explodedIndexEntryThreshold": "4000",
 "exportThreadCount": "0",
 "externalTxnDefaultBackendLockBehavior": "acquire-before-
retries",
 "externalTxnDefaultMaxLockTimeout": "100 ms",
 "externalTxnDefaultMinLockTimeout": "100 ms",
 "externalTxnDefaultRetryAttempts": "2",
 "hashEntries": "false",
 "importTempDirectory": "import-tmp",
 "importThreadCount": "16",
 "indexEntryLimit": "4000",
 "isPrivateBackend": "false",
 "javaClass":
 "com.unboundid.directory.server.backends.jeb.BackendImpl",
 "jeProperty": [  "\"je.env.backgroundReadLimit=0\""
 ],
 "numRecentChanges": "50000",
 "offlineProcessDatabaseOpenTimeout": "1 h",
 "primeAllIndexes": "true",
 "primeMethod": [
   "none"
 ],
 "primeThreadCount": "2",
 "primeTimeLimit": "0 ms",
 "processFiltersWithUndefinedAttributeTypes": "false",
 "returnUnavailableForUntrustedIndex": "true",
 "returnUnavailableWhenDisabled": "true",
 "setDegradedAlertForUntrustedIndex": "true",
 "setDegradedAlertWhenDisabled": "true",
 "subtreeDeleteBatchSize": "5000",
 "subtreeDeleteSizeLimit": "5000",
 "uncachedId2entryCacheMode": "cache-keys-only",
 "writabilityMode": "enabled",
 "urn:unboundid:schemas:configuration:messages:2.0": {
   "requiredActions": [
     {
       "property": "jeProperty",
       "type": "componentRestart",
       "synopsis": "In order for this modification to take effect,
          the component must be restarted, either by disabling and
          re-enabling it, or by restarting the server"
     },
     {
       "property": "id2childrenIndexEntryLimit",
       "type": "other",
       "synopsis": "If this limit is increased, then the contents
          of the backend must be exported to LDIF and re-imported
 to
          allow the new limit to be used for any id2children keys
          that had already hit the previous limit."
     }
   ]
 }
```

```
}
```

Configuration API paths

The Configuration API and supported sub-paths are available under the `/config` path.

A full listing of supported sub-paths is available when you access the base `/config/ResourceTypes` endpoint.

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

Some paths reflect hierarchical relationships between objects. For example, properties of a local DB VLV index for the `userRoot` backend are available using a path like `/config/backends/userRoot/local-db-indexes/uid`. Some paths represent singleton objects, which have properties but cannot be deleted or created. These paths can be differentiated from others by their singular, rather than plural, relation name, such as `global-configuration`.

The following sample response is abbreviated.

```
{
  "schemas": [
    "urn:ietf:params:scim:api:messages:2.0:ListResponse"
  ],
  "totalResults": 520,
  "Resources": [
    {
      "schemas": [

 "urn:ietf:params:scim:schemas:core:2.0:ResourceType"
      ],
      "id": "dsee-compat-access-control-handler",
      "name": "DSEE Compat Access Control Handler",
      "description": "The DSEE Compat Access Control
              Handler provides an implementation that uses
 syntax
              compatible with the Sun Java System Directory
 Server
              Enterprise Edition access control handler.",
      "endpoint": "/access-control-handler",
      "meta": {
        "resourceType": "ResourceType",
        "location": "http://example.com:5033/config/
ResourceTypes/dsee-compat-access-control-handler"
      }
    },
    {
      "schemas": [

 "urn:ietf:params:scim:schemas:core:2.0:ResourceType"
      ],
      "id": "access-control-handler",
      "name": "Access Control Handler",
      "description": "Access Control Handlers manage the
              application-wide access control. The server's
 access
              control handler is defined through an extensible
              interface, so that alternate implementations can
 be created.
```

```
               Only one access control handler may be active in
 the server
                          at any given time.",
      "endpoint": "/access-control-handler",
      "meta": {
        "resourceType": "ResourceType",
        "location": "http://example.com:5033/config/
ResourceTypes/access-control-handler"
      }
    },
    {
 ...
```

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

Sort and filter objects

The Configuration API supports System for Cross-domain Identity Management (SCIM) parameters for filter, sorting, and pagination.

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

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

Update properties

The Configuration API supports the HTTP PUT method as an alternative to modifying objects with HTTP PATCH.

With PUT, the server computes the differences between the object in the request with the current version in the server and performs modifications where necessary. The server never removes attributes that are not specified in the request. The API responds with the entire modified object.

Sample request:

```
PUT /config/backends/userRoot
Host: example.com:5033
Accept: application/scim+json
{
  "description" : "A new description."
```

```
}
```

Sample response:

```
{
  "schemas": [
    "urn:unboundid:schemas:configuration:2.0:backend:local-db"
  ],
  "id": "userRoot",
  "meta": {
    "resourceType": "Local DB Backend",
    "location": "http://example.com:5033/config/backends/
userRoot"
  },
  "backendID": "userRoot",
  "backgroundPrime": "false",
  "backupFilePermissions": "700",
  "baseDN": [
    "dc=example,dc=com"
  ],
  "checkpointOnCloseCount": "2",
  "cleanerThreadWaitTime": "120000",
  "compressEntries": "false",
  "continuePrimeAfterCacheFull": "false",
  "dbBackgroundSyncInterval": "1 s",
  "dbCachePercent": "25",
  "dbCacheSize": "0 b",
  "dbCheckpointerBytesInterval": "20 mb",
  "dbCheckpointerHighPriority": "false",
  "dbCheckpointerWakeupInterval": "30 s",
  "dbCleanOnExplicitGC": "false",
  "dbCleanerMinUtilization": "75",
  "dbCompactKeyPrefixes": "true",
  "dbDirectory": "db",
  "dbDirectoryPermissions": "700",
  "dbEvictorCriticalPercentage": "5",
  "dbEvictorLruOnly": "false",
  "dbEvictorNodesPerScan": "10",
  "dbFileCacheSize": "1000",
  "dbImportCachePercent": "60",
  "dbLogFileMax": "50 mb",
  "dbLoggingFileHandlerOn": "true",
  "dbLoggingLevel": "CONFIG",
  "dbNumCleanerThreads": "1",
  "dbNumLockTables": "0",
  "dbRunCleaner": "true",
  "dbTxnNoSync": "false",
  "dbTxnWriteNoSync": "true",
  "dbUseThreadLocalHandles": "true",
  "deadlockRetryLimit": "10",
  "defaultCacheMode":
  "cache-keys-and-values",
  "defaultTxnMaxLockTimeout": "10 s",
  "defaultTxnMinLockTimeout": "10 s",
  "description": "abc",
  "enabled": "true",
  "explodedIndexEntryThreshold": "4000",
  "exportThreadCount": "0",
  "externalTxnDefaultBackendLockBehavior":
  "acquire-before-retries",
  "externalTxnDefaultMaxLockTimeout": "100 ms",
  "externalTxnDefaultMinLockTimeout": "100 ms",
  "externalTxnDefaultRetryAttempts": "2",
```

```
  "hashEntries": "true",
  "importTempDirectory": "import-tmp",
  "importThreadCount": "16",
  "indexEntryLimit": "4000",
  "isPrivateBackend": "false",
  "javaClass":
 "com.unboundid.directory.server.backends.jeb.BackendImpl",
  "numRecentChanges":
 "50000",  "offlineProcessDatabaseOpenTimeout": "1 h",
  "primeAllIndexes": "true",
  "primeMethod": [
    "none"
  ],
  "primeThreadCount": "2",
  "primeTimeLimit": "0 ms",
  "processFiltersWithUndefinedAttributeTypes": "false",
  "returnUnavailableForUntrustedIndex": "true",
  "returnUnavailableWhenDisabled": "true",
  "setDegradedAlertForUntrustedIndex": "true",
  "setDegradedAlertWhenDisabled": "true",
  "subtreeDeleteBatchSize": "5000",
  "subtreeDeleteSizeLimit": "100000",
  "uncachedId2entryCacheMode": "cache-keys-only",
  "writabilityMode": "enabled"
}
```

Administrative actions

Updating a property might require an administrative action before changes can take effect.

If you need administrative actions to update a property, the server returns `200 Success`. Any actions are returned in the `urn:unboundid:schemas:configuration:messages:2.0` section of the JSON response that represents the entire object that was created or modified.

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

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

```
  }
```

Updating servers and server groups
You can configure servers as part of a server group so that configuration changes applied to a single server are applied to all servers in a group.

When managing a server that is a member of a server group, creating or updating objects using the Configuration API requires the `applyChangeTo` query attribute. The behavior and acceptable values for this parameter are identical to the `dsconfig` parameter of the same name. A value of `single-server` or `server-group` can be specified.

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

Configuration API responses
Clients of the API should examine the HTTP response code to determine the success or failure of a request.

The following are response codes and their meanings.

Response Code	Description	Response Body
200 Success	The requested operation succeeded, with the response body being the configuration object that was created or modified. If further actions are required, they are included in the `urn:unboundid:schemas:configuration:messages:2.0` object.	List of objects, object properties, or administrative actions
204 No Content	The requested operation succeeded and no further information has been provided, as in the case of a DELETE operation.	None
400 Bad Request	The request contents are incorrectly formatted or a request is made for an invalid API version.	Error summary and optional message
401 Unauthorized	User authentication is required. Some user agents, such as browsers, might respond by prompting for credentials. If the request specified credentials in an Authorization header, they are invalid.	None
403 Forbidden	The requested operation is forbidden, either because the user does not have sufficient privileges or some other constraint, such as an object is edit-only and cannot be deleted.	None
404 Not Found	The requested path does not refer to an existing object or object relation.	Error summary and optional message

Response Code	Description	Response Body
409 Conflict	The requested operation could not be performed due to the current state of the configuration. For example, an attempt was made to create an object that already exists, or an attempt was made to delete an object that is referenced by another object.	Error summary and optional message
415 Unsupported Media Type	The request is such that the Accept header does not indicate that JSON is an acceptable format for a response.	None
500 Server Error	The server encountered an unexpected error. Report server errors to Customer Support.	Error summary and optional message

ⓘ **Note:**

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

The following is an example encoded error message.

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

Working with the Directory REST API

The Directory REST API is the native interface for client access to the PingDirectory Server. Instead of trying to manage directory hierarchy or require attribute mapping, the Directory REST API provides direct access to directory data in a way that is dynamic, discoverable, and efficient.

Before you begin

The Directory REST API gives developers who are more comfortable with REST than LDAP access to arbitrary directory data in a way that ensures directory data remains consistent regardless of whether it is accessed from LDAP or REST. The Directory API is enabled during server setup. After setup, individual services and applications can be enabled or disabled by configuring the HTTPS Connection Handler.

While both the Directory REST API and System for Cross-domain Identity Management (SCIM) provide REST access to directory data, the goals of the two protocols are different. SCIM is useful to generic, external clients that require simple, narrow access to identity data, but because it is a less common standard for identity stores, it might not offer as much functionality or be as user-friendly as the Directory REST API.

The Directory REST API can be used for the following operations.

HTTP operation	Resource endpoint	Description	Allowed query parameters
DELETE	**/directory/v1/{dn}**	Delete an entry.	
GET	**/directory/v1**	Get metadata about the API and server.	
GET	**/directory/v1/{dn}**	Retrieve a single entry.	▪ expand ▪ includeAttributes ▪ excludeAttributes
GET	**/directory/v1/{dn}/subtree**	Search an entry's descendants.	▪ filter ▪ searchScope ▪ cursor ▪ limit ▪ includeAttributes ▪ excludeAttributes
GET	**/directory/v1/schemas**	Retrieve the schemas of all available object classes.	
GET	**/directory/v1/schemas/{objectclass}**	Retrieve schema for a specific object class.	
GET	**/directory/v1/schemas/_operationalAttributes**	Retrieve schema for operational attributes.	
GET	**/directory/v1/me**	Alias for retrieving the current user.	
PATCH	**/directory/v1/{dn}**	Modify an entry (add or delete values).	expand
POST	**/directory/v1**	Create a new entry.	expand
PUT	**/directory/v1/{dn}**	Modify or rename an entry.	expand

Steps

- Configure the Directory REST API with any of the following properties using **dsconfig**:

Command	Description
basic-auth-enabled	Specifies whether users can connect to the service with HTTP Basic authentication. If disabled, users need a bearer token. If changed, the server must be restarted, or any HTTP Connection Handlers referencing this service disabled and re-enabled. Basic authentication is enabled by default.
identity-mapper	If HTTP Basic authentication is enabled, the identity mapper referenced by this distinguished name (DN) must be used to map the

Command	Description
	user names provided to user entries. By default, an identity mapper is provided, which maps a fully-qualified DN to an entry. For changes to take effect, the server must be restarted, or any HTTP connection handlers referencing this service disabled and re-enabled.
access-token-validator	Specifies the subset of this server's Access Token Validators (by DN), which can validate Bearer authentication tokens. By default, if no validators are specified, then any of the validators on the server can be used. For changes to take effect, the server must be restarted, or any HTTP Connection Handlers referencing this service disabled and re-enabled.
access-token-scope	The scope that must be present in Bearer tokens in order to be accepted by this service. If no value is provided, Bearer token authentication is disabled, and only Basic authentication is used. By default, no value is provided. Changes to this value take effect immediately.
audience	A string or URI audience that must be present in Bearer tokens in order to be accepted by this service. If no value is provided, any audience is acceptable. By default, no value is provided. Changes to this value take effect immediately.
max-page-size	The maximum number of entries to be returned in one page from the search endpoint. Actual results returned might be lower due to the limit query parameter on the request and the actual number of available results. The value must be an integer between 1 and 1000. The default value is 100. Changes to this value take effect immediately.
schemas-endpoint-objectclass	The list of object classes that will be returned by the **/schemas/** endpoint in the REST API. By default, no schemas are returned. Changes to this value take effect immediately.

The following example uses **dsconfig** to configure an **objectClass** entity.

```
dsconfig set-http-servlet-extension-props --extension-name "Directory REST
 API" \
  --add schemas-endpoint-objectclass:ubidPerson
```

Configuring the Server using the Administrative Console

The PingDirectory Server provides an administrative console for server configuration and monitoring that has the same functionality as the **dsconfig** command.

When signing on to the Administrative Console, the console does not persistently store any credentials for authenticating to the Directory Server. Instead, the console uses the credentials provided by the user when signing on.

When managing multiple directory server instances, the provided credentials must be valid for each instance.

Signing on to the Administrative Console

About this task

To sign on to the console, enter a fully qualified distinguished name (DN), for example, `cn=admin2,cn=Topology Admin Users,cn=Topology,cn=config`. For more information, see *Configuring a global administrator* on page 387.

Steps

1. Start the Directory Server.

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

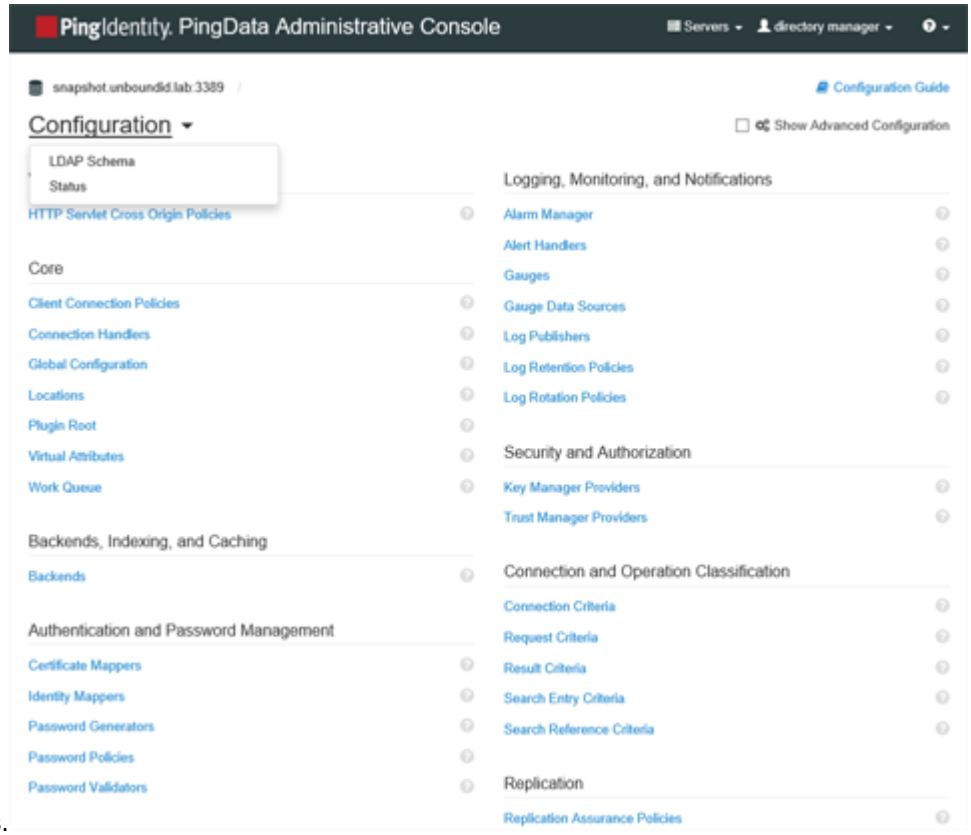
2. In a browser, go to *http://server-name:389/console/login*.
3. In the **Username** and **Password** fields, enter the root user DN credentials.

4. Click **Sign In**.

Configuring the Server using the Console

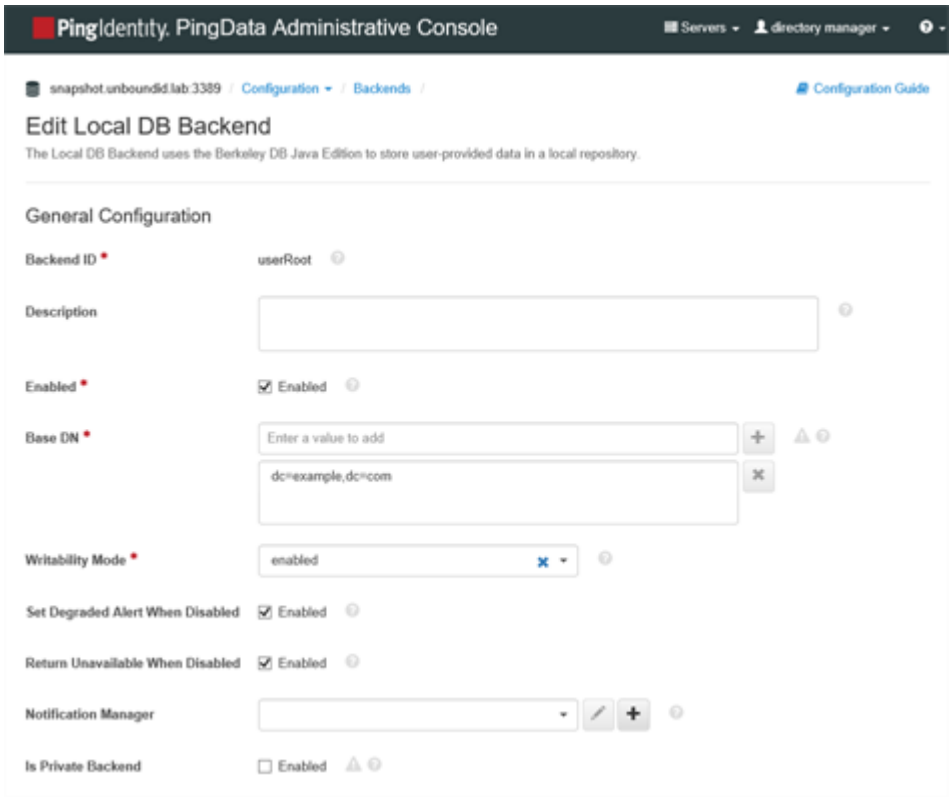
Steps

1. From the Administrative Console, go to **Configuration**#



   Backends.
2. At the top of the **Administrative Console** page, select the **Show Advanced Properties** check box.
3. Click the name of the backend you want to configure.

4. Change or enter values in the server configuration.



5. To apply your changes, click **Save**.

Generating a summary of configuration components

The `config-diff` toolgenerates a summary of the configuration in a local or remote directory server instance.

About this task

The `config-diff` tool is useful for comparing configuration settings on the directory server instance when troubleshooting issues or when verifying configuration settings on newly-added servers. The tool can interact with the local configuration regardless of whether the server is running.

Steps

- To generate a summary of configuration components, run `config-diff`
  `$ bin/config-diff` generates a summary of a local online server.
- To generate a comparison of the current configuration with the pre-installation configuration, run the following.

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

  This comparison ignores any changes that could be made by the installer and writes the output to a file called `configuration-steps.dsconfig`.

  You can use this file as the basis for a script to configure a new server identical to the local server.
- To view other available tool options, run `config-diff --help`.

Administrator account classes

Directory Server provides three different classes of administrator accounts: root user, administrator, and global administrator.

Root user

The root user is the LDAP-equivalent of a UNIX super-user account and inherits its privileges from the default root user privilege set. For more information on default root privileges, see *Default root privileges*. The root user account is an entry that is stored in the server's configuration under `cn=Root DNs,cn=config` and bypasses access control evaluation. It can be created manually or with the `dsconfig` tool. This account has full access to the entire set of data in the directory information tree (DIT) and to the server configuration and its operations. One important difference between other vendors' servers and Directory Server's implementation is that the root user's rights are granted through a set of privileges. This allows Directory Server to have multiple root users on its system, but the normal practice is to set up administrator user entries. The root user has no resource limits by default.

Administrator

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

Global administrator

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

Managing root user accounts

PingDirectoryProxy Server provides a default root user, `cn=Directory Manager`, that is stored in the server's configuration file, such as under `cn=Root DNs,cn=config`.

About this task

The root user is the LDAP-equivalent of a UNIX superuser account and inherits its read-write privileges from the default root privilege set.

Steps

- To create or update root users, use the `dscconfig` tool.

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

> ⓘ **Note:**
>
> Root user entries are stored in the server's configuration.

- To limit full access to all of Directory Proxy Server, create separate administrator accounts with limited privileges so that you can identify the administrator responsible for a particular change.

> ⓘ **Note:**
>
> Separate user accounts for each administrator make it possible to enable password policy functionality, such as password expiration, password history, and requiring secure authentication, for each administrator.

Default root privileges

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

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

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

Default Root Privileges

Privilege	Description
audit-data-security	Allows the associated user to execute data security auditing tasks.
backend-backup	Allows the user to perform backend backup operations.
backend-restore	Allows the user to perform backend restore operations.
bypass-acl	Allows the user to bypass access control evaluation.
config-read	Allows the user to read the server configuration.
config-write	Allows the user to update the server configuration.
disconnect-client	Allows the user to terminate arbitrary client connections.
ldif-export	Allows the user to perform LDIF export operations.
ldif-import	Allows the user to perform LDIF import operations.
lockdown-mode	Allows the user to request a server lockdown.
manage-topology	Allows the user to modify topology setting.
metrics-read	Allows the user to read server metrics.
modify-acl	Allows the user to modify access control rules.
password-reset	Allows the user to reset user passwords but not their own. The user must also have privileges granted by access control to write the user password to the target entry.

Privilege	Description
permit-get-password-policy-state-issues	Allows the user to access password policy state issues.
privilege-change	Allows the user to change the set of privileges for a specific user, or to change the set of privileges automatically assigned to a root user.
server-restart	Allows the user to request a server restart.
server-shutdown	Allows the user to request a server shutdown.
soft-delete-read	Allows the user access to soft-deleted entries.
stream-values	Allows the user to perform a stream values extended operation that obtains all entry DNs and/or all values for one or more attributes for a specified portion of the DIT.
third-party-task	Allows the associated user to invoke tasks created by third-party developers.
unindexed-search	Allows the user to perform an unindexed search in the Oracle Berkeley DB Java Edition backend.
update-schema	Allows the user to update the server schema.
use-admin-session	Allows the associated user to use an administrative session to request that operations be processed using a dedicated pool of worker threads.

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

Other Available Privileges

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

Configuring administrator accounts

An administrator account is any account in the user backend that is assigned one or more privileges or is given access to read and write operations beyond that of a normal user entry.

The privilege mechanism is the same as that used for root distinguished name (DN) accounts and allows individual privileges to be assigned to an administrator entry.

Typically, administrator user entries are controlled by access control evaluation to limit access to the entire set of data in the directory information tree (DIT). You can grant fine-grained read and write access using the access control definitions available through the `aci` attribute. Administrator entries reside in the backend configuration, for example, `uid=admin,dc=example,dc=com`, and are replicated between servers in a replication topology.

The following examples show how to configure administrator accounts:

- The first procedure shows how to set up a single, generic `uid=admin,dc=example,dc=com` account with limited privileges.

  > ⓘ **Note:**
  >
  > If you generated sample data at install, you can view an example `uid=admin` entry using **`ldapsearch`**.

- The second example shows a more realistic example where the user is part of the administrators group.

> ⓘ **Note:**
>
> Both examples are based on a simple DIT. Actual deployment cases depend on your schema.

Setting up a single administrator account

About this task

To create an example of a single, generic administrator account:

Steps

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

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

2. To add the entry, use the **`ldapmodify`** tool.

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

3. To add the access control instruction (ACI) to the root suffix or base DN to give full access to the new administrator, create another LDIF file.

   > ⓘ **Note:**
   >
   > The ACI grants full access to all user attributes, but not to operational attributes. To grant access to operational attributes as well as user attributes, use `(targetattr = "*||+")` in the access control instruction.

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

4. To add the entry, use the **`ldapmodify`** tool.

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

5. To verify the additions, use the **`ldapsearch`** tool.

   In the following example, the first command searches for the entry that contains `uid=Admin` and returns it if the search is successful. The second command searches for the base DN and returns only those operational attributes, including ACIs, associated with the entry.

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

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

6. Add specific privileges to the administrator account, then to process the modify operation press **CTRL-D**.

   For this example, add the `password-reset` privilege to the administrator account from the command line.

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

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

7. Assign a password policy for the administrator account.

   Create an `Admin Password Policy`, then add the password policy to the account.

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

8. To apply the password policy to the account, run the **`ldapmodify`** command.

   Execute the **`ldapmodify`** command with a bind DN that has sufficient rights, such as a root DN, as in the following example.

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

Changing the administrator password

About this task

Root users are governed by the root password policy and by default, their passwords never expire. To change a root user password, use the `ldappasswordmodify` tool.

Steps

1. Open a text editor and create a text file containing the new password.

   For this example, name the file `rootuser.txt`.

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

2. To change the root user's password, run **`ldappasswordmodify`**.

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

3. Remove the text file.

```
$ rm rootuser.txt
```

Setting up an administrator group

About this task

The following example shows how to set up a group of administrators that have access rights to the whole Directory Server.

> ⓘ **Note:**
>
> The example uses a static group using the `GroupOfUniqueNames` object class.

Steps

1. Create an LDIF file with an example administrator group.

   For this example, name the file `admin-group.ldif`

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

dn: cn=Dir Admins,ou=Groups,dc=example,dc=com
objectClass: groupofuniquenames
```

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

2. To add the entries, use the **ldapmodify** tool.

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

3. To add the ACI to the root suffix or base DN to provide full access to the Directory Server to the new administrator, create another LDIF file.

   For this example, name the file `admin-aci.ldif`.

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

4. To add the ACI, use the **ldapmodify** tool.

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

5. To verify the additions, use the **ldapsearch** tool.

   In the following example, the first command searches for the entry that contains `cn=Dir Admins` and returns it if the search is successful. The second command searches for the base DN and returns only those operational attributes, including ACIs, associated with the entry.

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

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

6. To add specific privileges to each administrator account, use an LDIF file.

   For this example, name the file `admin-priv.ldif`.

   For this example, add the `password-reset` privilege to the `user.0` administrator account from the command line. To add the privilege, use the **ldapmodify** tool. Repeat the process for the other administrators configured in the administrator group.

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

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

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

7. To assign a password policy for the administrator account, use an LDIF file. Save the file as `admin-pwd-policy.ldif`.

   For example, create an `Admin Password Policy`, then add the password policy to the account. To apply the password policy to the account, use the **`ldapmodify`** tool.

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

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

Configuring a global administrator

A global administrator is created when replication is enabled and is responsible for managing configuration server groups.

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

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

Creating a global administrator

Steps

1. To create a new global administrator, use the `create-topology-admin-user` option with **`dsconfig`**.

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

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

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

Removing a global administrator

Steps

1. To delete a global administrator, use the `delete-topology-admin-user` option with **`dsconfig`**.

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

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

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

Configuring server groups

The PingDirectory Server provides a mechanism for setting up administrative domains that synchronize configuration changes among servers in a server group.

About this task

After you have set up a server group, you can make an update on one server using **`dsconfig`**, then apply the change to the other servers in the group using the `--applyChangeTo server-group` option of the **`dsconfig`** non-interactive command. If you want to apply the change to one server in the group, use the `--applyChangeTo single-server` option. When using **`dsconfig`** in interactive command-line mode, you are asked if you want to apply the change to a single server or to all servers in the server group.

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

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

Steps

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

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

2. Add any directory server to the server group.

   If you have set up replication between a set of servers, these server entries are created by the **`dsreplication`** `enable` command.

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

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

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

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

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

4. Test the server group.

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

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

```
  --applyChangeTo server-group
```

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

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

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

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

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

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

   The first directory server instance should be disabled.

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

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

9. View the property on the second directory server instance.

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

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

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

Client connection policy configuration

Client connection policies help distinguish what portions of the DIT the client can access.

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

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

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

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

About the client connection policy

Client connection policies are based on two factors.

Connection criteria

The connection criteria are used in many areas within the server. They are used by the client connection policies, but they can be used in other instances when the server needs to perform matching based on connection-level properties, such as filtered logging. A single connection can match multiple connection criteria definitions.

Evaluation order index

If multiple client connection policies are defined in the server, then each of them must have a unique value for the `evaluation-order-index` property. The client connection policies are evaluated in order of ascending evaluation order index. If a client connection does not match the criteria for any defined client connection policy, then that connection will be terminated.

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

When a client connection policy is assigned

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

- When the connection is initially established. This association occurs exactly once for each client connection.
- After completing processing for a StartTLS operation. This association occurs once at most for a client connection because StartTLS cannot be used more than once on a particular connection. You can not stop using StartTLS while keeping the connection active.
- After completing processing for a bind operation. This association occurs zero or more times for a client connection because the bind request can be processed many times on a given connection.

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

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

Restricting the type of search filter used by clients

You can restrict the types of search filters that a given client might be allowed to use to prevent the use of potentially expensive filters, like range or substring searches.

You can use the `allowed-filter-type` property to provide a list of filter types that can be included in the search requests from clients associated with the client connection policy. This setting is only used if search is included in the set of allowed operation types. This restriction only applies to searches with a scope other than `baseObject`, such as searches with a scope of `singleLevel`, `wholeSubtree`, or `subordinateSubtree`.

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

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

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

Resource limits

Client connection policies can specify resource limits, helping to ensure that no single client monopolizes server resources.

You can limit the total number of connections to a server from a particular client or from clients that match specified criteria. You can also limit the duration of the connection.

A client connection policy can only be used to enforce additional restrictions on a client connection. You cannot use it to grant a client capabilities that it would not otherwise have.

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

Resource Limiting Properties

Property	Description
`maximum-concurrent-connections`	Specifies the maximum number of client connections that can be associated with that client connection policy at any given time. The default value of zero indicates that no limit is enforced. If the server already has the maximum number of connections associated with a client connection policy, then any attempt to associate another connection with that policy, such as newly-established connections or an existing connection that has done something to change its client connection policy, such as perform a bind or StartTLS operation, causes that connection to be terminated.

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

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

Property	Description
`allowed-filter-type`	Specifies the types of components that might be used in filters included in search operations with a non-base scope that are requested by clients associated with the client connection policy. Any non-base scoped search request whose filter contains a component not included in this set is rejected. The set of possible filter types include: ▪ and ▪ or ▪ not ▪ equality ▪ sub-initial ▪ sub-any ▪ sub-final ▪ greater-or-equal ▪ less-or-equal ▪ approximate-match ▪ extensible-match By default, all filter types are allowed. ⓘ **Note:** No restriction is placed on the types of filters that might be used in searches with a base scope.
`allow-unindexed-searches`	Specifies whether clients associated with the client connection policy are allowed to request searches that cannot be efficiently processed using the configured set of indexes. ⓘ **Note:** Clients must still have the `unindexed-search` privilege, so this option does not grant the ability to perform unindexed searches to clients that would not have otherwise had that ability, but it might be used to prevent clients associated with the client connection policy from requesting unindexed searches when they might have otherwise been allowed to do so. By default, this has a value of "true", indicating that any client associated with the client connection policy that has the `unindexed-search` privilege is allowed to request unindexed searches.
`minimum-substring-length`	Specifies the minimum number of bytes, which might be present in any sub- Initial, subAny, or subFinal element of a substring search filter component in a search with a non-baseObject scope. A value of one (which is the default) indicates that no limit is enforced. This property might be used to prevent clients from issuing overly-vague substring searches that might require installing the Directory Server to examine too many entries over the course of processing the request.

Property	Description
`maximum-search-size-limit`	Specifies the maximum number of entries that might be returned from any single search operation requested by a client associated with this client connection policy. **Note:** This property only specifies a maximum limit and never increases any limit that might already be in effect for the client thought the `size-limit` global configuration property or the `ds-rlim-size-limit` operational attribute. A value of zero (default) indicates that no additional limit is enforced on top of whatever size limit might already be in effect for an associated connection. If a nonzero value is provided, then the effective maximum size limit for any search operation requested by the client is the smaller of the size limit from that search request, the `maximum-search-size-limit` from the client connection policy, and the size limit that would otherwise be in effect for that client.

Defining the operation rate

Configure the maximum operation rate for individual client connections and collectively for all connections associated with a client connection policy.

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

Specify rate limit strings as a maximum count, followed by a slash and a duration. The count portion must contain an integer and can be followed by the following multipliers:

- k (to indicate that the integer should be interpreted as thousands)
- m (to indicate that the integer should be interpreted as millions)
- g (to indicate that the integer should be interpreted as billions)

The duration portion must contain a time unit of milliseconds (ms), seconds (s), minutes (m), hours (h), days (d), or weeks (w) and can be preceded by an integer to specify a quantity for that unit.

For example, the following are valid rate limit strings:

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

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

Client connection policy deployment example

In this example scenario, we assume the following:

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

For more information on this client connection policy deployment example, see the following topics:

Define the connection policies
Set a per-client connection policy limit on the number of connections that can be associated with a particular client connection policy

Define at least two client connection policies, one for each of the two clients. Each policy must have different connection criteria for selecting the policy with which a given client connection should be associated.

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

Define the following three client connection policies:

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

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

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

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

How the policy is evaluated
Whenever a connection is established, the server associates the connection with exactly one client connection policy.

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

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

index. If these criteria do not match, the server finally checks the connection criteria associated with the Unauthenticated Connection Policy because it has the third lowest evaluation order index. It finds a match, so the client connection is associated with the Unauthenticated Connection Policy.

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

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

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

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

Configuring a client connection policy using the console
Configure a client connection policy using the PingDirectory Administrative Console.

Steps

1. From the Administrative Console, in the **Core Server** section, click **Client Connection Policies**.

   > ⓘ **Tip:**
   >
   > If you do not see **Client Connection Policies** on the menu, change the **Object Types** filter to **Standard**.

2. To add a new policy, click **Add New**.
3. Enter a **Policy ID**.

   If you want to base your new client connection policy on an existing policy, select it from the **Template** menu.
4. Configure the properties of the client connection policy, then to enable the policy, select **Enabled**.
5. Enter the order in which you want the new policy to be evaluated in the **Evaluation Order Index** box, and then click **Continue**.

   A policy with a lower index is evaluated before a policy with a higher index. The Directory Server uses the first evaluated policy that applies to a client connection.

6. Select the connection criteria that match the client connection for this policy.
   a. To change the criteria, click **View** and **Edit**.
   b. To add new criteria, click **Select New**.
   c. Select the operations allowed for clients that are members of this connection group.
   d. To make operations available to clients, use the **Add** and **Remove** buttons.
   e. Specify the extended operations that clients are allowed and denied to use.
7. Enter the type of authorization allowed and the SASL mechanisms that are allowed and denied in response to client requests.
8. Select the **Include Backend Subtree Views** check box if you want to automatically include the subtree views of backends configured in the Directory Server.

   You can also choose to include and exclude specific base DNs using the appropriate fields.
9. Save your changes.
   - To review the **`dsconfig`** command equivalent and save your changes, click **Confirm then Save**.
   - To save your changes without first reviewing the **`dsconfig`** output, click **Save Now**.

Configuring a client connection policy using dsconfig
Configure a client connection policy using the **`dsconfig`** tool.

About this task

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

Steps

1. Use the **`dsconfig`** tool to create and configure a client connection policy.

   Specify the host name, connection method, port number, and bind distinguished names (DNs) described in previous procedures.
2. To change to the **Standard objects** menu, from the Directory Server main menu, enter `o`, and then enter the number for the **Standard** menu.
3. From the Directory Server main menu, enter the number associated with **Client Connection Policy**.
4. From the **Client Connection Policy management** menu, enter the number corresponding to **Create a new connection policy**.
5. To create a new client connection policy from scratch, enter `n`.
6. Enter a name for the new client connection policy.
7. To enable the connection policy, from the **Enabled Property** menu, select **True**.
8. To set the evaluation order for the policy, from the **Evaluation-Order Property** menu, enter a value between 0 and 2147483647.

   A client connection policy with a lower evaluation-order is evaluated before one with a higher number. For this example, enter `9999`.
9. From the **Client Connection Policy** menu, review the configuration.
   a. If you want to make any further modifications, enter the number corresponding to the property.
   b. To finish the creation of the client connection policy, enter `f`.

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

Restricting server access based on client IP address
Two methods are available to limit client access to the Directory Server.

Connection Handlers

You can limit the IP addresses using the LDAP or LDAPS connection handlers. The connection handlers provide an `allowed-client` property and a `denied-client` property. The `allowed-client` property specifies the set of allowable address masks that can establish connections to the handler. The `denied-client` property specifies the set of address masks that are not allowed to establish connections to the handler.

Client Connection Policies

For a more fine-grained approach, restrict access by configuring a new client connection policy. Then, create a new connection criteria and associate it with the connection policy. A connection criteria defines sets of criteria for grouping and describing client connections based on a number of properties, including the protocol, client address, connection security, and authentication state for the connection. Each client connection policy can be associated with zero or more connection criteria. Server components can use connection criteria to indicate which connections to process and what kind of processing to perform, such as to select connections and operations for filtered logging or to classify connections for network groups.

Restricting server access using the connection handlers
Use **`dsconfig`** to set the `allowed-client` property for the LDAP connection handler.

Steps

- Set the `allowed-client` property for the LDAP connection handler using **`dsconfig`**.
- Specify the address mask for the range of allowable IP addresses that can establish connections to the Directory Server.
- To configure the server using the **`dsconfig`** tool on the local host, specify the loopback address to 127.0.0.1.

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

Restricting server access using client connection policies
A client-established connection to the PingDirectory server is associated with a client connection policy. Use client connection policies to restrict the kinds of requests that the client can issue and impose resource limits for that connection.

Steps

1. Create a simple connection criteria.

   The following example uses the **`dsconfig`** tool in non-interactive mode. It allows only the Directory Server's IP address and loopback to have access.

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

2. Assign the criteria to the client connection policy.

   ```
   $ bin/dsconfig set-client-connection-policy-prop \
     --policy-name new-policy \
   ```

```
  --set connection-criteria:allowed-ip-addrs
```

After you have run the command, access is denied to remote IP addresses. The Directory Server does not require a restart.

3. Add a remote IP range to the criteria.

> ⓘ **Note:**
>
> For the following example, add `10.6.1.*`.

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

Access from any remote servers is allowed. The Directory Server does not require a restart.

4. To restore default behavior, remove the criteria from the connection policy.

> ⓘ **Tip:**
>
> Include the LDAP or LDAPS connection parameters, such as host name, port, bindDN, bindPassword, with the **`dsconfig`** command.

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

The Directory Server does not require a restart.

Automatically authenticating clients that have a secure communication channel

The Directory Server provides the option to automatically authenticate clients that have a secure communication channel, either SSL or StartTLS, and to present their own certificate.

About this task

By default, this option is disabled. When enabled, the net effect is as if the client issued a SASL EXTERNAL bind request on that connection.

> ⓘ **Note:**
>
> This option is ignored if the client connection is already authenticated, such as when using StartTLS, but the client had already performed a bind before the StartTLS request. If the bind attempt fails, the connection remains unauthenticated but usable. If the client subsequently sends a bind request on the connection, it's processed as normal, and any automatic authentication is destroyed.

Steps

- Run the following **`dsconfig`** command.

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

Securing the Server with lockdown mode

The Directory Server provides tools to enter and leave lockdown mode if the server requires a security lockdown.

About this task

In lockdown mode, only users with the `lockdown-mode` privilege can perform operations Users who do not have the privilege are rejected. By default, root users have this privilege. You can give other administrators this privilege. Users with this privilege can configure lockdown mode as a recurring task.

Some configuration problems can lead to inadvertent exposure of sensitive information, such as an access control rule that cannot be properly parsed, and cause the server to place itself in lockdown mode. This ensures that an administrator can manually correct the problem. Lockdown mode does not persist across restarts.

Steps

- To perform some administrative operations and ensure that other client requests are not allowed to access any data in the server, manually place the Directory Server into lockdown mode.

  Any client request to the Directory Server in lockdown mode receives an `Unavailable` response.
- To take the directory server out of lockdown mode, use either of the following options:
  - Use the **`leave-lockdown-mode`** command.
  - Restart the server.
- To start a server in lockdown mode, use the **`start-server`** `--lockdownMode` option.

Entering lockdown mode manually

Steps

- To enter lockdown mode, run **`enter-lockdown-mode`**.

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

Leaving lockdown mode

Steps

- To leave lockdown mode, run **`leave-lockdown-mode`**.

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

Starting a server in lockdown mode

Steps

- To start a server in lockdown mode, run the **`start-server`** command with the `--lockdownMode` option.

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

Configuring maximum shutdown time

The Directory Server provides an option for administrators to set a maximum time for a shutdown process to take.

Before you begin

During shutdown, some database checkpointing and cleaning threads might remain active even after the default time period on systems with large or busy database backends. If checkpointing or cleaning is aborted prematurely, it can lead to significantly longer startup times for the Directory Server. When a shutdown process is initiated, the server begins stopping all of its internal components and waits up to 5 minutes for all threads to complete before exiting.

About this task

Use the **`dsconfig`** tool to configure the maximum shutdown time to allow database operations to complete.

Steps

- To increase the maximum shutdown time for your system, choose from one of the following options.
  - Run the **`dsconfig`** tool.

    ⓘ **Note:**

    The following command increases the maximum shutdown time from 5 minutes to 6 minutes. The command allows time values of w (weeks), d (days), h (hours), m (minutes), s (seconds), ms (milliseconds).

    Include the LDAP or LDAPS connection parameters, such as host name, port, bindDN, and bindDN password, with the **`dsconfig`** command.

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

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

About working with referrals

A referral is a redirection mechanism that tells client applications that a requested entry or set of entries is not present on the Directory Server but can be accessed on another server.

Use referrals when entries are located on another server. The Directory Server implements two types of referrals depending on the requirement:

Referral on Update plugin

The Directory Server provides a Referral on Update plugin to create any referrals for update requests, such as `add`, `delete`, `modify`, or `moddn` operations, on read-only servers. For example, given two replicated directory servers where one server is a master with read-write capabilities, and the other is a read-only server. You can configure a referral for any client update requests on the second directory server to point to the master server. For example, if a client application sends an add request on the second directory server, the directory server responds with a referral that indicates any updates should be made on the master server. All read requests on the read-only server process normally.

Smart referrals

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

Specifying LDAP URLs

Referrals use LDAP URLs to redirect a client application's request to another server.

LDAP URLs have a specific format, described in RFC 4516 and require that all special characters be properly escaped and any spaces indicated as "%20". LDAP URLs have the following syntax.

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

`ldap[s]`

Indicates the type of LDAP connection to the Directory Server. If the Directory Server connects over a standard, non-encrypted connection, then LDAP is used. If it connects over SSL, then LDAPS is used.

> ⓘ **Note:**
>
> Any search request initiated by means of an LDAP URL is anonymous by default unless an LDAP client provides authentication.

`hostname`

Specifies the host name or IP address of the Directory Server.

`port`

Specifies the port number of the Directory Server. If no port number is provided, the default LDAP port (389) or LDAPS port (636) is used.

`base-dn`

Specifies the distinguished name (DN) of an entry in the directory information tree (DIT). The Directory Server uses the base DN as the starting point entry for its searches. If no base DN is provided, the search begins at the root of the DIT.

`attributes`

Specifies those attributes for which the Directory Server should search and return. You can indicate more than one attribute by providing a comma-separated list of attributes. If no attributes are provided, the search returns all attributes.

`scope`

Specifies the scope of the search, which could be one of the following:

`base`

Only searches the specified base DN entry.

`one`

Only search one level below the specified base DN.

`sub`

Searches the base entry and all entries below the specified base DN.

If no scope is provided, the server performs a base search.

`filter`

Specifies the search filter to apply to entries within the scope of the search. If no filter is provided, the server uses `+`.

Creating referrals
Create a smart referral by adding an entry with the referral and `extensibleObject` object classes or adding the object classes to a specific entry.

About this task

The referral object class designates the entry as a referral object. The `extensibleObject` object class allows you to match the target entry by matching any schema attribute. The following example shows how to set up a smart referral if a portion of a directory information tree (DIT) is located on another server.

To create a referral:

Steps

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

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

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

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

3. To verify the addition, search for a user.

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

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

Modifying a referral
Run a modify command to modify a referral.

Steps

- To modify the `ref` attribute on the referral entry, run **`ldapmodify`** with the `manageDSAIT` control.

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

Deleting a referral
Run `ldapdelete` to delete a referral.

Steps

- To delete the referral entry, run **`ldapdelete`** with the `manageDSAIT` control.

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

Configuring a read-only server

The PingDirectory Server provides a means to configure a hub-like, read-only directory server for legacy systems that require it.

About this task

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

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

Steps

1. Install two replicating directory servers.

   For more information on various ways to set up your servers, see *Enabling Replication*.
2. On the second server, run the **`dsconfig`** command to set the writability mode of the server to internal-only.

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

3. On the second server, run the **`dsconfig`** command to create a referral that instructs the server to redirect client write requests under `dc=example,dc=com` to server1.example.com:1389.

   The referral itself is defined as a plugin of type `referral-on-update`. This command sets up the server to process read operations but redirects all write operations under `dc=example,dc=com` to another server.

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

4. To test the referral, attempt to modify an entry and confirm that the server responds with the result code of 10.

   The resulting message is available in the server's access log.

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

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

Configuring HTTP access for the Directory Server

Although most clients communicate with the PingDirectory Server using LDAP, the server also provides support for an HTTP connection handler that uses Java servlets to serve content to clients over HTTP.

Ping Identity offers an extension that uses this HTTP connection handler to add support for the System for Cross-domain Identity Management (SCIM) protocol. Third-party developers can also use the Server SDK to write extensions that leverage this HTTP support.

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

For more information, see the following topics:

- *Configuring HTTP Servlet Extensions* on page 406
- *Configuring HTTP operation loggers* on page 407
- *Example HTTP log publishers* on page 408
- *Configuring HTTP connection handlers* on page 410

Configuring HTTP Servlet Extensions

You must first configure one or more servlet extensions to use the HTTP connection handler.

Servlet extensions are responsible for obtaining Java servlets, using the Java Servlet 3.0 specification as described in JSR 315, and registering them to be invoked using one or more context paths. If you plan to deploy the System for Cross-domain Identity Management (SCIM) extension, follow the instructions in *Managing the SCIM 2.0 Servlet Extension* on page 1195. For custom servlet extensions created using the Server SDK, the process varies based on whether you are using a Java-based or Groovy-scripted extension.

Configuring web application servlet extensions

About this task

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

Steps

- When deploying a web application from a directory, specify the location of the `web.xml` deployment descriptor if it is not in the standard location.
- Specify the directory used by the server for temporary files.

Results
At runtime, the web application has access to the server classes.

Configuring Java-based servlet extensions

About this task

For Java-based extensions, first use the Server SDK to create and build the extension bundle as described in the Server SDK documentation.

Steps

- Use the **`manage-extension`** tool to install it.

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

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

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

Configuring Groovy-scripted extensions

Steps

1. For Groovy-scripted extensions, place the necessary Groovy scripts in the appropriate directory based on the package for those scripts after the `lib/groovy-scripted-extensions` directory.
2. Create a new Groovy Scripted HTTP Servlet extension, specifying the fully-qualified Groovy class name for the `script-class` property, and providing any appropriate arguments in the `script-argument` property.

Configuring HTTP operation loggers
Interactions between servlet extensions and HTTP clients are generally not recorded in the server access log unless a servlet extension performs internal operations to interact with data held in the directory server.

About this task

To capture information about communication with HTTP clients, you must configure one or more HTTP operations loggers.

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

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

The log message contains the following elements:

- The IP address of the client that issued the request

- The RFC 1413 (ident) identity of the client

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

Steps

- To configure an HTTP operation logger to use this common log format, create a new instance of a Common Log File HTTP Operation Log Publisher object.
  a. Specify the path and name for the active log file.
  b. Specify the rotation and retention policies to manage the log files.

  > ⓘ **Note:**
  >
  > Properties for Common Log File HTTP Operation Log Publisher objects generally have the same meaning and use as they do for other kinds of loggers.
- To create custom Java-based or Groovy-scripted HTTP operation loggers using the Third Party HTTP Operation Log Publisher and Groovy Scripted HTTP Operation Log Publisher object types, use the Server SDK.

Example HTTP log publishers
When troubleshooting HTTP connection handler issues, start with the logs to determine any potential problems.

About this task

The following examples walk you through how to use various **`dsconfig`** commands and their associated options to configure log files to use for troubleshooting connection handler issues.

The following section shows some **`dsconfig`** commands and their corresponding log files.

Steps

- To configure a default detailed HTTP Log Publisher with default log rotation and retention policies, use the `create-log-publisher` option with **`dsconfig`**.

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

```
  --set "retention-policy:Size Limit Retention Policy"
```

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

- The time that the request was received
- The request ID issued by the client, including the IP address, port, HTTP method, and URL
- The authorization type, request content type, and status code
- The response content length
- The redirect URI
- The response content type

The HTTP log file shows the following.

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

- To configure a detailed HTTP Log Publisher with request and response header names and values, including the `Content-Type` request header, use the `set-log-publisher-prop` option with **`dsconfig`**.

> ⓘ **Note:**
>
> The `Content-Type` request header is suppressed by default.

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

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

- The request header for the host, HTTP method, content type, connection, user agent
- The response header for the access-control credentials

```
[23/Oct/2012:11:39:41-0600] RESULT requestID=4665307 from="10.5.0.20:56044"
method="GET" url="https://10.5.1.129:443/Beth/Users?
attributes=userName,title,
emails,urn:scim:schemas:extension:custom:1.0:descriptions,urn:scim:schemas:
extension:enterprise:1.0:manager,groups,urn:scim:schemas:extension:custom:1.0:
blob&filter=userName+eq+%22user.18935%22" requestHeader="Host:
 x2270-11.example.
lab:443" requestHeader="Accept: application/json" requestHeader="Content-
Type:
```

```
application/json" requestHeader="Connection: keep-alive"
requestHeader="User-Agent: Wink Client v1.1.2" authorizationType="Basic"
requestContentType="application/json" statusCode=200 etime=140.384
responseContentLength=11778 responseHeader="Access-Control-Allow-
Credentials:
true" responseContentType="application/json"
```

Another log example shows an example user creation event. The client is curl.

```
[23/Oct/2016:11:50:11-0600] RESULT requestID=4802791 from="10.8.1.229:52357"
method="POST" url="https://10.2.1.113:443/Aleph/Users/" requestHeader="Host:
 x2270-
11.example.lab" requestHeader="Expect: 100-continue" requestHeader="Accept:
 applica-
tion/xml" requestHeader="Content-Type: application/xml" requestHeader="User-
Agent:
curl/7.21.4 (universal-apple-darwin11.0) libcurl/7.21.4 OpenSSL/0.9.8r
 zlib/1.2.5"
authorizationType="Basic" requestContentType="application/xml"
 requestContent-
Length=1773 statusCode=201 etime=11.598 responseContentLength=1472 redirec-
tURI="https://x2270-11.example.lab:443/Aleph/Users/b2cef63c-5e46-11e1-974b-
60334b1a0d7a" responseContentType="application/xml"
```

The final example shows a user deletion request. The client is the Sync Server.

```
[23/Oct/2016:11:38:06-0600] RESULT requestID=4610261 from="10.5.1.114:34558"
method="DELETE" url="https://10.2.1.113:443/Aleph/Users/b8547525-24e0-41ae-
b66b-
0b441800de70" requestHeader="Host: x2270-11.example.lab:443"
 requestHeader="Accept:
application/json" requestHeader="Content-Type: application/json"
 requestHeader="Con-
nection: keep-alive" requestHeader="User-Agent: DataSync-6.0.0.0 (Build
20121022173845Z, Revision 11281)" authorizationType="Basic"
 requestContentType="appli-
cation/json" statusCode=200 etime=10.615 responseContentLength=0
```

Configuring HTTP connection handlers

HTTP connection handlers are responsible for managing the communication with HTTP clients and invoking servlets to process requests from those clients. HTTP connection handlers can be used to host web applications on the server.

Each HTTP connection handler should be configured with one or more HTTP servlet extensions and zero or more HTTP operation log publishers.

If the HTTP Connection Handler cannot be started (for example, if its associated HTTP Servlet Extension fails to initialize), then this does not prevent the entire Directory Server from starting. Directory Server's **`start-server`** tool outputs any errors to the error log. This allows Directory Server to continue serving LDAP requests even with a bad servlet extension.

The configuration properties available for use with an HTTP connection handler include the following:

listen-address

Specifies the address on which the connection handler listens for requests from clients. If not specified, then requests are accepted on all addresses bound to the system.

listen-port

Specifies the port on which the connection handler listens for requests from clients. Required.

use-ssl

Indicates whether the connection handler uses SSL/TLS to secure communications with clients, such as whether it uses HTTPS rather than HTTP. If SSL is enabled, then `key-manager-provider` and `trust-manager-provider` values must also be specified.

http-servlet-extension

Specifies the set of servlet extensions that are enabled for use with the connection handler. You can have multiple HTTP connection handlers listening on different address/port combinations with identical or different sets of servlet extensions. You must configure at least one servlet extension.

http-operation-log-publisher

Specifies the set of HTTP operation log publishers that should be used with the connection handler. By default, no HTTP operation log publishers are used.

key-manager-provider

Specifies the key manager provider that is used to obtain the certificate presented to clients if SSL is enabled.

trust-manager-provider

Specifies the trust manager provider that is used to determine whether to accept any client certificates presented to the server.

num-request-handlers

Specifies the number of threads that should be used to process requests from HTTP clients. These threads are separate from the worker threads used to process other kinds of requests. The default value of zero means the number of threads are automatically selected based on the number of CPUs available to the Java virtual machine (JVM).

web-application-extension

Specifies the web applications hosted by the server.

Configuring an HTTP connection handler

About this task

An HTTP connection handler has two dependent configuration objects:

- One or more HTTP servlet extensions
- An HTTP log publisher

  The log publisher is optional, but in most cases, you should configure one or more logs to troubleshoot any issues with your HTTP connection.

> ⓘ **Note:**
>
> You must configure the HTTP servlet extension and log publisher before configuring the HTTP connection handler.

Steps

1. To configure your HTTP servlet extensions, use the `create-http-servlet-extension` option with **`dsconfig`**.

   This example uses the `ExampleHTTPServletExtension` example in the Server SDK.

   ```
   $ bin/dsconfig create-http-servlet-extension \
     --extension-name "Hello World Servlet" \
     --type third-party \
   ```

```
  --set "extension-
class:com.unboundid.directory.sdk.examples.ExampleHTTPServletExtension" \
  --set "extension-argument:path=/" \
  --set "extension-argument:name=example-servlet"
```

2. To configure one or more HTTP log publishers, use the `create-log-publisher` option with **`dsconfig`**.

   This example configures two log publishers: one for common access and one for detailed access. Both log publishers use the default configuration settings for log rotation and retention.

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

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

3. To configure the HTTP connection handler, specify the HTTP servlet extension and log publishers using the `create-connection-handler` option with **`dsconfig`**.

   > ⓘ **Note:**
   >
   > You can update some configuration properties later as needed, but for others, like `listen-port`, you must disable and then re-enable the HTTP Connection Handler for the change to take effect.

```
$ bin/dsconfig create-connection-handler \
  --handler-name "Hello World HTTP Connection Handler" \
  --type http \
  --set enabled:true \
  --set listen-port:8443 \
  --set use-ssl:true \
  --set "http-servlet-extension:Hello World Servlet" \
  --set "http-operation-log-publisher:HTTP Common Access Logger" \
  --set "http-operation-log-publisher:HTTP Detailed Access Logger" \
  --set "key-manager-provider:JKS" \
  --set "trust-manager-provider:JKS"
```

4. To monitor the connection handler, use the **`ldapsearch`** tool.

   > ⓘ **Note:**
   >
   > By default, the HTTP connection handler has an advanced monitor entry property, `keep-stats`, that is set to `TRUE` by default.

```
$ bin/ldapsearch --baseDN "cn=monitor" \
  "(objectClass=ds-http-connection-handler-statistics-monitor-entry)"
```

Configuring an HTTP connection handler for web applications

About this task

To host web applications on the server, use HTTP connection handlers.

Steps

1. To create the web application servlet extension, use the `create-web-application-extension` option with **`dsconfig`**.

```
$ bin/dsconfig create-web-application-extension \
  --extension-name "Hello Web Application" \
  --set "base-context-path:/hello-app" \
  --set "document-root-directory:/opt/hello-web-app"
```

2. To monitor the connection handler, use the **`ldapsearch`** tool.

> ⓘ **Note:**
>
> By default, the HTTP connection handler has an advanced monitor entry property, `keep-stats`, that is set to `TRUE`.

```
$ bin/ldapsearch --baseDN "cn=monitor" \
  "(objectClass=ds-http-connection-handler-statistics-monitor-entry)"
```

HTTP correlation IDs
Correlation IDs make it easier to track log messages across a software system request that passes through multiple subsystems.

Scattered and intermingled log messages create challenges for tracing the request flow on distributed systems. A correlation ID can be assigned to a request and added to all associated operations as the request is processed. A correlation ID allows related log messages to be easily located and grouped. The server supports correlation IDs for all HTTP requests received through its HTTP or HTTPS Connection Handler.

When an HTTP request is received, it is automatically assigned a correlation ID. This ID can be used to correlate HTTP responses with messages recorded to the HTTP Detailed Operation log and the trace log. For specific web APIs, the correlation ID might also be passed to the LDAP subsystem.

The correlation ID appears with associated requests in LDAP logs in the **`correlationID`** key for the following REST APIs:

- SCIM 1
- Delegated admin
- Consent
- Directory

The correlation ID is used as the default client request ID value in intermediate client request controls used by the following REST APIs:

- SCIM 2
- Consent
- Directory

Values related to the intermediate client request control appear in the LDAP logs in the **`via`** key and are forwarded to downstream LDAP servers when received by PingDirectoryProxy Server. The correlation ID header is also added to requests forwarded by the PingDataGovernance gateway.

For Server SDK extensions that have access to the current `HttpServletRequest`, the current correlation ID can be retrieved as a string through the `HttpServletRequest` **`com.pingidentity.pingdata.correlation_id`** attribute, as shown.

```
(String) request.getAttribute("com.pingidentity.pingdata.correlation_id");
```

Configuring HTTP correlation ID support

About this task

Correlation ID support is enabled by default for each HTTP Connection Handler.

Steps

- To enable or disable correlation ID support for the HTTPS Connection Handler, use the `set-connection-handler-prop` option with **`dsconfig`**.

  This example shows how to enable correlation ID support.

  ```
  $ dsconfig set-connection-handler-prop \
    --handler-name "HTTPS Connection Handler" \
    --set use-correlation-id-header:true
  ```

  This example shows how to disable correlation ID support.

  ```
  $ dsconfig set-connection-handler-prop \
    --handler-name "HTTPS Connection Handler" \
    --set use-correlation-id-header:false
  ```

- To customize the response header name for the correlation ID, use the `set-connection-handler-prop` option with **`dsconfig`**.

  > ⓘ **Note:**
  >
  > The server generates a correlation ID for every HTTP request and sends it in the response through the **`Correlation-Id`** response header.

  This example changes the **`correlation-id-response-header`** property value to `X-Request-Id`.

  ```
  $ dsconfig set-connection-handler-prop \
     --handler-name "HTTPS Connection Handler" \
     --set correlation-id-response-header:X-Request-Id
  ```

- To designate the names of one or more HTTP request headers that contain an existing correlation ID value, use the `set-connection-handler-prop` option with **`dsconfig`**.

  > ⓘ **Note:**
  >
  > This enables the server to integrate with a larger system consisting of every servers using correlation IDs.
  >
  > By default, the server generates a new, unique correlation ID for each HTTP request and ignores any correlation ID that might be set on the request.

  ```
  $ dsconfig set-connection-handler-prop --handler-name "HTTPS Connection
   Handler" \
    --set correlation-id-request-header:X-Request-Id \
    --set correlation-id-request-header:X-Correlation-Id \
    --set correlation-id-request-header:Correlation-Id \
  ```

```
   --set correlation-id-request-header:X-Amzn-Trace-Id
```

HTTP correlation ID example

In this example, a request to the Directory REST API is made and the correlation ID enables finding HTTP-specific log messages with LDAP-specific log messages. The response to the API call includes a Correlation-Id header with the value **`a54aee33-c6c6-4467-be25-efd1db7a8b76`**.

```
GET /directory/v1/me?includeAttributes=mail HTTP/1.1
 Accept: */*
 Accept-Encoding: gzip, deflate
 Authorization: Bearer ...
 Connection: keep-alive
 Host: localhost:1443
 User-Agent: HTTPie/0.9.9

 HTTP/1.1 200 OK
 Content-Length: 266
 Content-Type: application/hal+json
 Correlation-Id: ee919049-6710-4594-9c66-28b4ada4b127
 Date: Fri, 02 Nov 2018 15:16:50 GMT
 Request-Id: 369

 {
      "_dn": "uid=user.86,ou=People,dc=example,dc=com",
      "_links": {
         "schemas": [
            {
              "href": "https://localhost:1443/directory/v1/schemas/
inetOrgPerson"
            }
         ],
         "self": {
              "href": "https://localhost:1443/directory/v1/
uid=user.86,ou=People,dc=example,dc=com"
         }
     },
     "mail": [
         "user.86@example.com"
     ]
 }
```

This correlation ID can be used to search the HTTP trace log for matching log records, as follows:

```
$  grep 'correlationID="ee919049-6710-4594-9c66-28b4ada4b127"'
 PingDirectory/logs/debug-trace
      [02/Nov/2018:10:16:50.294 -0500] HTTP REQUEST requestID=369
 correlationID="ee919049-6710-4594-9c66-28b4ada4b127" product="PingDirectory
 Directory Server" instanceName="ds1" startupID="W9ikqA==" threadID=52358
 from=[0:0:0:0:0:0:0:1]:58918 method=GET url="https://0:0:0:0:0:0:0:1:1443/
directory/v1/me?includeAttributes=mail"
 [02/Nov/2018:10:16:50.526 -0500] DEBUG ACCESS-TOKEN-VALIDATOR-PROCESSING
 requestID=369 correlationID="ee919049-6710-4594-9c66-28b4ada4b127"
 msg="Identity Mapper with DN 'cn=User ID Identity Mapper,cn=Identity
 Mappers,cn=config' mapped ID 'user.86' to entry DN
 'uid=user.86,ou=people,dc=example,dc=com'"
 [02/Nov/2018:10:16:50.526 -0500] DEBUG ACCESS-TOKEN-VALIDATOR-PROCESSING
 requestID=369 correlationID="ee919049-6710-4594-9c66-28b4ada4b127"
 accessTokenId="201811020831" msg="Token Validator 'Mock Access Token
 Validator' validated access token with active = 'true', sub = 'user.86',
 owner = 'uid=user.86,ou=people,dc=example,dc=com', clientId = 'client1',
 scopes = 'ds', expiration = 'none', not-used-before = 'none', current time
 = 'Nov 2, 2018 10:16:50 AM CDT' "
```

```
 [02/Nov/2018:10:16:50.531 -0500] HTTP RESPONSE requestID=369
 correlationID="ee919049-6710-4594-9c66-28b4ada4b127"
 accessTokenId="201811020831" product="PingDirectory Directory Server"
 instanceName="ds1" startupID="W9ikqA==" threadID=52358 statusCode=200
 etime=236.932 responseContentLength=266
      [02/Nov/2018:10:16:50.531 -0500] DEBUG HTTP-FULL-REQUEST-AND-RESPONSE
 requestID=369 correlationID="ee919049-6710-4594-9c66-28b4ada4b127"
 accessTokenId="201811020831" product="PingDirectory Directory
 Server" instanceName="ds1" startupID="W9ikqA==" threadID=52358
 from=[0:0:0:0:0:0:0:1]:58918 method=GET url="https://0:0:0:0:0:0:0:1:1443/
directory/v1/me?includeAttributes=mail" statusCode=200 etime=236.932
 responseContentLength=266 msg="
```

The LDAP log messages associated with this request can also be located:

```
$  grep 'correlationID="ee919049-6710-4594-9c66-28b4ada4b127"'
 PingDirectory/logs/access
 [02/Nov/2018:10:16:50.529 -0500] SEARCH RESULT instanceName="ds1"
 threadID=52358 conn=-371045 op=1657393 msgID=1657394
 origin="Directory REST API" httpRequestID="369"
 correlationID="ee919049-6710-4594-9c66-28b4ada4b127"
 authDN="uid=user.86,ou=people,dc=example,dc=com" requesterIP="internal"
 requesterDN="uid=user.86,ou=People,dc=example,dc=com"
 requestControls="1.3.6.1.4.1.30221.2.5.2" via="app='PingDirectory-
ds1' clientIP='0:0:0:0:0:0:0:1' sessionID='201811020831'
 requestID='ee919049-6710-4594-9c66-28b4ada4b127'"
 base="uid=user.86,ou=people,dc=example,dc=com" scope=0 filter="(&)"
 attrs="mail,objectClass" resultCode=0 resultCodeName="Success" etime=0.684
 entriesReturned=1
 [02/Nov/2018:10:16:50.530 -0500] EXTENDED RESULT
 instanceName="ds1" threadID=52358 conn=-371046 op=1657394
 msgID=1657395 origin="Directory REST API" httpRequestID="369"
 correlationID="ee919049-6710-4594-9c66-28b4ada4b127"
 authDN="cn=Internal Client,cn=Internal,cn=Root DNs,cn=config"
 requesterIP="internal" requesterDN="cn=Internal Client,cn=Internal,cn=Root
 DNs,cn=config" requestControls="1.3.6.1.4.1.30221.2.5.2"
 via="app='PingDirectory-ds1' clientIP='0:0:0:0:0:0:0:1'
 sessionID='201811020831' requestID='ee919049-6710-4594-9c66-28b4ada4b127'"
 requestOID="1.3.6.1.4.1.30221.1.6.1" requestType="Password
 Policy State" resultCode=0 resultCodeName="Success"
 etime=0.542 usedPrivileges="bypass-acl,password-reset"
 responseOID="1.3.6.1.4.1.30221.1.6.1" responseType="Password Policy State"
 dn="uid=user.86,ou=People,dc=example,dc=com"
 [02/Nov/2018:10:16:50.530 -0500] SEARCH RESULT instanceName="ds1"
 threadID=52358 conn=-371048 op=1657397 msgID=1657398
 origin="Directory REST API" httpRequestID="369"
 correlationID="ee919049-6710-4594-9c66-28b4ada4b127" authDN="cn=Internal
 Client,cn=Internal,cn=Root DNs,cn=config" requesterIP="internal"
 requesterDN="cn=Internal Client,cn=Internal,cn=Root DNs,cn=config"
 requestControls="1.3.6.1.4.1.30221.2.5.2" via="app='PingDirectory-
ds1' clientIP='0:0:0:0:0:0:0:1' sessionID='201811020831'
 requestID='ee919049-6710-4594-9c66-28b4ada4b127'" base="cn=Default Password
 Policy,cn=Password Policies,cn=config" scope=0 filter="(&)" attrs="ds-
cfg-password-attribute" resultCode=0 resultCodeName="Success" etime=0.065
 preAuthZUsedPrivileges="bypass-acl,config-read" entriesReturned=1
```

DNS caching

You can use two global configuration properties to control the caching of host name-to-numeric IP address or DNS lookup results returned from the name resolution services of the underlying operating system.

About this task

Steps

1. To configure these properties, use the **`dsconfig`** tool.

   Global configuration properties and their descriptions

Global configuration property	Description
network-address-cache-ttl	Sets the Java system property **`networkaddress.cache.ttl`**, and controls the length of time in seconds that a host name-to-IP address mapping can be cached. By default, it keeps resolution results for one hour (3600 seconds). This setting applies to the server and all extensions loaded by the server.
network-address-outage-cache-enabled	Caches host name-to-IP address results in the event of a DNS outage. By default, this is set to true, meaning name resolution results are cached. Unexpected service interruptions might occur during planned or unplanned maintenance, network outages, or an infrastructure attack. This cache can allow the server to function during a DNS outage with minimal impact. This cache is not available to server extensions.

2. To reduce delays due to unnecessary DNS lookups, follow these recommendations:
   a. Maintain a connection pool in the client app rather than opening new connections for each bind.
   b. Add appropriate records to DNS, including PTR records.
   c. Add `options timeout:1` or `options single-request` in the `/etc/resolv.conf` file.
   d. If IPv6 requests are causing issues, add `-Djava.net.preferIPv4Stack=true` to the `start-server.java-args` line in PingDirectory Server's `config/java.properties` file, so that running `bin/dsjavaproperties` and restarting the server no longer issues IPv6 PTR requests.

IP address reverse name lookups

The PingDirectory Server performs numeric IP address-to-host name lookups.

Numeric IP address-to-host name lookups

The following are some of the numeric IP address-to-host name lookups:

- Binding to the Directory: Decoding, examining, or evaluating a DNS bind rule
- Logging: Logging information to certain monitors or writing to the error log
- Java Management Extension (JMX): Creating a server socket
- Key Management: Generating a trust store
- Replication Server: Creating an SSL socket
- Replication Session Management: Obtaining a session or performing a handshake with a replication server
- Simple Authentication and Security Layer (SASL) Authentication: Applying configuration changes
- SMTP Alert Handler: Initializing or sending an alert notification

Configuring address masks

Address masks configured in Access control instructions (ACIs), connection handlers, connection criteria, and certificate handshake processing might trigger implicit reverse name lookups.

For more information about how address masks are configured in the server, see the following information for each server:

- ACI DNS: bind rules in *Managing Access Control* on page 566 for Directory Server and Directory Proxy Server
- `ds-auth-allowed-address`: *Restricting access through operational attributes in user entities* for Directory Server
- Connection Criteria: *Restricting server access based on client IP address* on page 399 for Directory Server and Directory Proxy Server
- Connection Handlers: *Restricting server access using the connection handlers* on page 399 for a configuration reference guide for all servers

Configuring traffic through a load balancer

Configure the PingDirectory Server to get traffic through a load balancer and to record the actual client's IP address.

Before you begin

By default, when a PingDirectory Server is sitting behind an intermediate HTTP server, such as a load balancer, a reverse proxy, or a cache, it logs incoming requests as originating with the intermediate HTTP server instead of the client that sent the request.

Steps

1. To record the actual client's IP address to the trace log, enable **`X-Forwarded-*`** handling in both the intermediate HTTP server and the PingDirectory Server.
2. Edit the appropriate connection handler object, HTTPS or HTTP, and set **`use-forwarded-headers`** to **`true`**.

   > ⓘ **Note:**
   >
   > When **`use-forwarded-headers`** is set to **`true`**, the server uses the client IP address and port information in the **`X-Forwarded-*`** headers instead of the address and port of the entity that's sending the request, the load balancer. This client address information shows up in logs, such as in the **`from`** field of the HTTP REQUEST and HTTP RESPONSE messages.

3. On the load balancer, configure settings to provide the **`X-Forwarded-*`** information, such as **`X-Forwarded-Host:`**.

Working with the Referential Integrity plugin

Referential Integrity is a plugin mechanism that maintains the distinguished name (DN) references between an entry and a group member attribute. If you have a group entry consisting of member attributes specifying the DNs of printers, you can enable the referential integrity plugin to ensure that the group entry is automatically removed if a printer entry is removed from the Directory Server.

Before you begin

By default, the Referential Integrity plugin is disabled. When enabled, the plugin performs integrity updates on the specified attributes, such as `member` or `uniquemember`, after a delete, modify DN, or a rename, such as subordinate `modifyDN`, operation is logged to the `logs/ referint` file. If an entry is deleted, the plugin checks the log file and makes the corresponding change to the associated group entry.

Important points about the Referential Integrity plugin:

- Index all specified attributes that are configured for Referential Integrity.
- On replicated servers, the Referential Integrity plugin configuration is not propagated to other replicas. You must manually enable the plugin on each replica.

- The plugin settings must be identical on all machines.
- If the Referential Integrity plugin is enabled and configured to operate in synchronous mode, subtree delete operations are not allowed. You must configure the plugin to operate in asynchronous mode by specifying a nonzero update interval for subtree delete operations to perform.

About this task

Enable the Referential Integrity plugin.

Steps

1. Determine the attributes needed for your system.

   By default, the `member` and the `uniquemember` attributes are set for the plugin.
2. To enable the Referential Integrity plugin, run the **`dsconfig`** tool.

```
$ bin/dsconfig set-plugin-prop --plugin-name "Referential Integrity" \
  --set enabled:true
```

Working with the Unique Attribute plugin

The Unique Attribute plugin enforces uniqueness constraints on the values of one or more attributes across a portion of the Directory Server. The plugin checks for uniqueness prior to an add, modify, or modify distinguished name (DN) request and instructs the server to reject the request if a constraint violation is found.

About this task

By default, the plugin is disabled because it can affect performance in heavy write load environments. After the plugin is enabled, it does not check for attribute uniqueness on existing entries, only on new `ADD`, `MODIFY`, or `MODDN` operations. To ensure that no such conflicts exist in the data, administrators can use the **`identify-unique-attribute-conflicts`** command.

> ⓘ **Important:**
>
> Ensure all attributes to enforce for uniqueness are indexed for equality in all backends.
> Use the LDAP SDK uniqueness request control for enforcing uniqueness on a per-request basis. For more information on the LDAP SDK documentation and the `com.unboundid.ldap.sdk.unboundidds.controls.UniquenessResponseControl` class for using the control, see *Use the server SDK and LDAP SDK*. See the ASN.1 specification to implement support for it in other APIs.

You can enforce attribute uniqueness in replicated environments in which each replica contains the complete set of data for which to provide uniqueness, regardless of whether clients communicate directly with the server or interact with it through a Directory Proxy Server. In such environments, all servers have identical uniqueness configurations.

> ⓘ **Note:**
>
> It is not possible to prevent conflicts that arise from simultaneous writes on separate replicas. However, such conflicts are detected after the changes have been replicated and then triggers administrative alert notifications.

For proxied environments that do not have the complete set of data on all servers, such as environments that use entry balancing or that store different portions of the DIT on different servers, implement the Global Uniqueness Attribute plugin on the Directory Proxy Server instead of enabling the attribute

uniqueness plugin on the PingDirectory Server Admin Guide. For more information, see the *PingDirectory Server Admin Guide* and the *PingDirectoryProxy Server Admin Guide*.

To enable the Unique Attribute plugin:

Steps

1. Determine which attributes must be unique in your data.
2. To enable the plugin, run the **dsconfig** tool.

   By default, the plugin type property is set to `postsynchronizationadd`, `postsynchronizationmodify`, `postsynchronizationmodifydn`, `preoperationadd`, `preoperationmodify`, and `preopertionmodifydn`.

   The following example checks for attribute uniqueness prior to `ADD` operation using the `--set plugin-type:preoperationadd` option.

   ```
   $ bin/dsconfig set-plugin-prop --plugin-name "UID Unique Attribute" \
     --set enabled:true
   ```

   a. If you want to set one plugin type, use the `--set plugin-type:<operation-type>` option.

Working with the Purge Expired Data plugin

Use the Purge Expired Data plugin to delete expired entries or attribute values and cleanup expired PingFederate Persistent Access Grants.

When the plugin is enabled, a background thread in the plugin periodically searches for and purges expired data. For optimal performance, enable the Purge Expired Data plugin on multiple servers in a topology. For example, you can configure one server to delete data while others are searching for expired data.

Create and configure the Purge Expired Data plugin with the **dsconfig** tool. Configuration options include the base distinguished name (DN) and filter, the items to be purged, how to identify expired data, and the frequency for polling and purging. You must index the search for expired data. An alarm is raised if the server purging data falls behind the configured **max-updates-per-second**. Monitoring information is available in the Admin Console, or **cn=monitor**.

Configuring the Purge Expired Data plugin for expired entries

About this task

Use the Purge Expired Data plugin to delete all unverified account entries that have not been accessed in the past eight weeks. The is useful for the following scenarios:

- Accounts that potential customers started to create through an application's registration process but then did not complete.
- The phone number or email address that was provided during registration was not verified and should be allowed to be used by another account.

Steps

1. If necessary, enable the Last Access Time plugin:

   The server can track the last access time automatically in the `ds-last-access-time` attribute by enabling the Last Access Time plugin.

   ```
   $ bin/dsconfig set-plugin-prop \
     --plugin-name "Last Access Time" \
     --set enabled:true
   ```

2. To determine expiration order, create an index on the date attribute.

   The Purge Expired Data plugin requires the date attribute that is used to determine expiration to be indexed for ordering.

```
$ bin/dsconfig create-local-db-index \
  --backend-name userRoot \
  --index-name ds-last-access-time \
  --set index-type:ordering
```

3. If there is data present in the directory, rebuild the index.

```
$ bin/rebuild-index \
  --baseDN dc=example,dc=com \
  --index ds-last-access-time
```

4. Create the plugin that purges account entries **`objectclass=account`** that are not verified.

   The following example purges account entries **`verified=false`** after eight weeks of inactivity.

```
$ bin/dsconfig create-plugin \
  --plugin-name "Purge Old Unvalidated Accounts" \
  --type purge-expired-data \
  --set enabled:true \
  --set datetime-attribute:ds-last-access-time \
  --set "expiration-offset:8 w" \
  --set "filter:(&(objectClass=account)(verified=false))"
```

Configuring the Purge Expired Data plugin for expired attribute values

About this task

Use the Purge Expired Data plugin to delete values of an attribute that have expired. For example, an application can track information about an employee's session and then expire the session after 24 hours. There can be multiple active sessions tracked across different devices with session information as shown in the following example.

In this example, the LDAP attribute is `sessioninfo` and the JSON field that stores the timestamp is `creationTime`. These are used to configure the Purge Expired Data plugin.

```
sessionInfo: { "sessionId" : "E85FAC04E331FFCA55549B10B7C7A4FA",
"ipAddress": "10.0.0.00", "userAgent": "Mozilla/5.0 (iPad; U; CPU OS 3_2
 like Mac OS X; en-us)
AppleWebKit/531.21.10 (KHTML, like Gecko) Version/4.0.4 Mobile/7B367
 Safari/531.21.10",
"creationTime" : "2018-03-31T13:10:15Z" }
```

Create the plugin to purge the JSON attribute values after 24 hours, rather than the entire session entry.

Steps

1. Create an index on the `creationTime` field of the `sessioninfo` attribute.

```
$ bin/dsconfig create-json-attribute-constraints \
  --attribute-type sessioninfo \
  --set enabled:true
```

```
$ bin/dsconfig create-json-field-constraints \
  --attribute-type sessioninfo \
  --json-field creationTime \
  --set index-values:true \
```

```
  --set value-type:string
```

2. Create and enable the plugin.

```
$ bin/dsconfig create-plugin \
  --plugin-name "Purge Old Session Data Plugin" \
  --type purge-expired-data \
  --set enabled:true \
  --set "custom-datetime-format:yyyy-MM-dd'T'HH:mm:ss'Z'" \
  --set datetime-attribute:sessioninfo \
  --set datetime-format:custom \
  --set datetime-json-field:creationTime \
  --set "expiration-offset:1 d" \
  --set purge-behavior:delete-json-attribute-values
```

Configuring uniqueness across attribute sets

Configure attribute uniqueness across a set of attributes using the `multiple-attribute-behavior` property.

About this task

To enable uniqueness across attribute sets:

Steps

- To configure the UID unique attribute plugin to apply across multiple attributes, use the `dsconfig` tool.

  The following table details the `multiple-attribute-behavior` property and the values it can take.

  The `multiple-attribute-behavior` property values, descriptions, and examples

Value	Description	Example
`unique-within-each-attribute`	If multiple attributes are specified, then uniqueness is enforced for all values of each attribute, but the same value might appear in different attributes in the same entry or in different entries.	For example, assume you have an existing entry that has attributes `telephoneNumber=123-456-7890` and `mobile=123-456-7891`. If you set the uniqueness plugin to have `--set "multiple-attribute-behavior:unique-within-each-attribute"` and add: ▪ An entry with a `telephoneNumber` value that matches the `telephoneNumber` attribute in the existing entry, then the add request fails. ▪ An entry with a `mobile` value that matches the `mobile` attribute in the existing entry, then the request fails. ▪ An entry with the same `telephoneNumber` and `mobile` attribute values, such as 123-456-7893, but differs from the values in the existing entry, then the add request succeeds.

Value	Description	Example
`unique-across-all-attributes-including-in-same-entry`	If multiple attributes are specified, then uniqueness is enforced across all of those attributes. If a value appears in one of those attributes that value might not be present in any other of the listed attributes in the same entry, nor in any of the listed attributes in other entries.	For example, assume you have an existing entry that has attributes, `telephoneNumber=123-456-7890` and `mobile=123-456-7891`. If you set the uniqueness plugin to have `--set "multiple-attribute-behavior:unique-across-all-attributes-including-in-same-entry"` and add: ▪ An entry with a `telephoneNumber` value, such as 123-456-7890, that matches the `telephoneNumber` attribute in an existing entry, then the add request fails. ▪ An entry with a `mobile` value that matches the `mobile` attribute in an existing entry, then the request fails. ▪ An entry with a `mobile` value, such as 123-456-7890, that matches the `telephoneNumber` attribute in an existing entry, then that fails. ▪ An entry with the same `telephoneNumber` and `mobile` attribute values, such as 123-456-7893, but differ from the values in an existing entry, then the add request fails.

Value	Description	Example
`unique-across-all-attributes-except-in-same-entry`	If multiple attributes are specified, then uniqueness is enforced across all of those attributes. So, if a value appears in one of those attributes, that value might not be present in any of the listed attributes in other entries. However, the same value might appear in multiple attributes in the same entry.	For example, assume you have an existing entry that has attributes, `telephoneNumber=123-456-7890` and `mobile=123-456-7891`. You set the uniqueness plugin to have `--set "multiple-attribute-behavior:unique-across-all-attributes-except-in-same-entry"` and add: ▪ An entry with a `telephoneNumber` value, such as 123-456-7890, that matches the `telephoneNumber` attribute in an existing entry, then the add request fails. ▪ An entry with a `mobile` value that matches the `mobile` attribute in the existing entry, then the request fails. ▪ An entry with a `mobile` value, such as 123-456-7890, that matches the `telephoneNumber` attribute in an existing entry, then the request fails. ▪ An entry with a `telephoneNumber` value, such as 123-456-7891, that matches the `mobile` attribute in an existing entry, then the request fails. ▪ An entry with the same `telephoneNumber` and `mobile` attribute values, such as 123-456-7893, but that differs from the values in an existing entry, then the add request succeeds.

The `multiple-attribute-behavior` property is set to `unique-within-each-attribute`, which indicates that uniqueness is enforced for all values of each attribute, such as `telephoneNumber=123-456-7890` and `mobile=123-456-7891`, but the same value, such as either 123-456-7890 or 123-456-7891 might appear in different attributes in the same entry or in different entries.

```
$ dsconfig create-plugin \
  --plug-in "Unique telephoneNumber and mobile" \
  --type "unique-attribute" \
  --set "enabled:true" \
  --set "type:telephoneNumber" \
```

```
  --set "type:mobile" \
  --set "multiple-attribute-behavior:unique-within-each-attribute" \
  --no-prompt
```

Working with the Last Access Time plugin

Use the Last Access Time plugin to record the timestamp of the last activity targeting an entry. The plugin updates the **ds-last-access-time** attribute of the entry when accessing it using an `ADD`, `BIND`, `COMPARE`, `MODIFY`, `MODIFY DN`, or `SEARCH` operation.

Before you begin

Consider the following before using this plugin:

- An updated **ds-last-access-time** attribute value is replicated like any other change to an entry.
- The **ds-last-access-time** attribute is not returned from a search, unless included in the attributes list explicitly, or given the "+" specification for operational attributes.
- The **ds-last-access-time** value format is **yyyyMMddHHmmss.SSS'Z'**, which provides millisecond-level accuracy, such as **20131207144135.821Z**.
- The **ds-last-access-time** attribute can be indexed with a local database index. The ordering index type is the most relevant, but might require a higher index entry limit (default is 4000) to accommodate searches for entries that are not accessed for a long period of time. The ordering index type with a short time range or high index entry limit results in indexed search results for requests, such as **(ds-last-access-time>=20131207144135.821Z)**.

About this task

Use the plugin with the Directory Server Uncached Attribute Criteria or any application that needs to determine the last access time of an entry. The plugin also enables defining request criteria to limit the scope of tracking the last access time. The **max-search-result-entries-to-update** property also prevents mass updates of **ds-last-access-time** when searches contain many results, but might not reflect end-user access.

Steps

- Enable the plugin on all servers that have the same configuration.

> ⓘ **Important:**
>
> For deployments earlier than version 4.5 that use the Last Access Time plugin, disable the plugin before upgrading, and then re-enable it after the update is complete. If servers are running different versions, the `last-access-time` updates might occur with a higher frequency than intended.

Working with the Pass-Through Authentication plugin

Use the Pass-Through Authentication plugin to delegate bind operations to remote LDAP servers by forwarding simple bind requests to an external LDAP server, including Active Directory.

Before you begin

Consider the following points before using the Pass-Through Authentication plugin:

- Configure the plugin to attempt a local bind, to set or update a local password, and to bypass local password policies to ensure remote passwords are migrated.
- Remote servers that accept a forwarded bind request might require connection security, such as a secure StartTLS or LDAPS TLS connection.
- Updating a password in PingDirectory Server might result in divergent passwords between the local and remote server. If necessary, use PingDataSync Server to synchronize passwords between servers.

The following table identifies and describes the configuration properties associated with the Pass-Through Authentication plugin.

The Pass-Through Authentication plugin properties and their descriptions

Property	Description
server-access-mode	Determines whether the servers are accessed in **round-robin**, **failover-on-unavailable**, or **failover-on-any-failure** mode. The default server access mode is **round-robin**.
update-local-password	Indicates whether the local password value requires updating to the value used in the bind request in the event that the local bind fails but the forwarded bind succeeds. To update passwords, a local entry must previously exist.
allow-lax-pass-through-authentication-passwords	Indicates whether updates to the local password value accept passwords that do not meet local password policy requirements.
connection-criteria	Specifies a set of connection criteria that must match the client associated with the local bind request for the bind to be forwarded to the remote server.
request-criteria	Specifies a set of request criteria that must match the local bind request or a local target entry for the bind to be forwarded to the remote server.
dn-map	Specifies one or more distinguished name (DN) mappings that can transform bind DNs before attempting to forward the bind to remote servers.
search-base-dn	Use when searching for a remote user entry by using a filter constructed from the pattern that the **search-filter-pattern** property defines. A DN map and search filter pattern cannot both be configured. If neither a DN map nor a search filter pattern is defined, user entries are expected to have the same DN in the local server and the remote servers.

Steps

- Enable the plugin on all servers that use the same configuration.

The Pass-Through Authentication plugin for the PingOne for Customers service
Although PingDataSync Server supports bidirectional synchronization between PingDirectory Server and the PingOne for Customers service, using the Pass-Through Authentication plugin for PingOne

for Customers Service can propagate password changes from the PingOne for Customers service into PingDirectory Server.

This plugin features a mandatory **try-local-bind** configuration property that enables one of the following modes of operation:

- When **try-local-bind** is `true`, the plugin attempts to authenticate locally first. It sends a request to the PingOne for Customers service only if the local bind attempt fails.
- When **try-local-bind** is `false`, the plugin attempts to authenticate with the PingOne for Customers service first.

The following table identifies and describes the configuration properties associated with the Pass-Through Authentication plugin for PingOne for Customers Service.

Pass-Through Authentication plugin properties and their descriptions for PingOne for Customers Service

Property	Description	Required	Default
api-url	URL that PingDirectory Server uses to communicate with the PingOne for Customers service.	Yes	N/A
auth-url	URL that PingDirectory Server uses to authenticate to the PingOne for Customers service.	Yes	N/A
oauth-client-id	OAuth client ID that PingDirectory Server uses to authenticate to the PingOne for Customers service.	Yes	N/A
oauth-client-secret	OAuth client secret that PingDirectory Server uses to authenticate to the PingOne for Customers service.	Yes	N/A
environment-id	Identifier for the PingOne for Customers environment that contains the users for whom pass-through authentication is attempted.	Yes	N/A
included-local-entry-base-dn	If this value is set, authentication attempts are passed to the PingOne for Customers service only for users in a specified DN. If this value is set, only users who exist within a specified base DN allow their authentication attempts to be passed through to the PingOne for Customers service.	No	All public naming contexts (if not set)
connection-criteria	Reference to a connection criteria object to use to identify the bind requests to pass-through to the PingOne for Customers service based on the server's knowledge of the client expected to be the address, protocol, and security level. If this property is defined, only client connections that match the criteria are included. If this property is not defined, all clients are included.	No	N/A

Property	Description	Required	Default
request-criteria	Reference to a request criteria object to use to identify the bind requests to pass through to the PingOne for Customers service, based on the contents of the request. If this property is defined, only bind requests that match the criteria are included. If this property is not defined, all bind requests are included.	No	N/A
try-local-bind	Indicates whether PingDirectory Server tries to process the bind locally before forwarding the bind request to the PingOne for Customers service. If this value is set to `true` and the bind succeeds locally, PingDirectory Server does not make a request to the PingOne for Customers service. If this value is set to `false`, PingDirectory Server ignores local credentials and attempts to authenticate only to the PingOne for Customers service.	Yes	`True`
override-local-password	Indicates whether PingDirectory Server attempts to bind to the PingOne for Customers service if the local account has a password. This property is used if **try-local-bind** is `true`. If it has a value of `false`, the plugin attempts to authenticate to the PingOne for Customers service only if the local user account does not have a password. If the local bind attempt fails while this value is set to `true`, the server tries to authenticate to the PingOne for Customers service even if the local account has a password.	Yes	`True`

Property	Description	Required	Default
update-local-password	Indicates whether PingDirectory Server attempts to set the password for the local user account, regardless of whether one is already set, when the local authentication attempt fails but the attempt to authenticate with the PingOne for Customers service succeeds. This property is used only if **try-local-bind** is `true`. If the on-premise PingDirectory Server is the authoritative source for passwords, set this property to `false` and configure PingDataSync Server to synchronize password changes from PingDirectory Server into the PingOne for Customers service. If the passwords differ, either the local password or the password for the PingOne for Customers service allows the user to authenticate. If the PingOne for Customers service is the authoritative source for passwords, set this property to `true`. To ensure that a pass-through attempt to the PingOne for Customers service does not override local changes, make all password changes in the PingOne for Customers service.	Yes	`False`
allow-lax-pass-through-authentication-passwords	Indicates whether PingDirectory Server bypasses the normal password-validation process when setting the local password from the PingOne for Customers service. This property is used only when both **try-local-bind** and **update-local-password** are `true`. If this value is `true` when a local bind attempt fails but the authentication attempt with the PingOne for Customers service succeeds, the user's password is updated locally even if a local attempt to change the password to the same value is rejected because the password is considered too weak. If this value is `false`, pass-through authentication succeeds if the authentication with PingOne for Customers service succeeds. However, the password for the PingOne for Customers service does not replace the local password.	Yes	`True`
ignored-password-policy-state-error-condition	Set of zero or more password policy state error conditions that are ignored for pass-through authentication. For a list of values and their descriptions, see the following table.	No	N/A

Property	Description	Required	Default
user-mapping-local-attribute	Name of an LDAP attribute that is used to map local user entries to the corresponding PingOne for Customers account. This property must include the same number of values as the **user-mapping-remote-json-field** property, and the order of their values is correlated. If multiple values are specified, all attributes must be present in the local entry, and the plugin performs an `AND` search in the PingOne for Customers service to locate the user account with all the values in the corresponding fields. The `entryDN` attribute can be used to represent the DN of the local entry.	Yes	N/A
user-mapping-remote-json-field	The name of a PingOne for Customers field used to map local user entries to the corresponding PingOne for Customers account. This property must include the same number and order of values as the `user-mapping-local-attribute` property.	Yes	N/A
additional-user-mapping-scim-filter	The System for Cross-domain Identity Management (SCIM) filter included in the search and used to identify the PingOne for Customers account that corresponds to the local user entry. If a value is provided for this property, it is used with the SCIM filter that was created to map the local user entry to a PingOne for Customers account. If a value is not provided for this property, no additional filter is used.	No	N/A

The following table identifies the values to use with the optional configuration property **ignored-password-policy-state-error-condition** and describes the scenarios in which a user is permitted to bind when using pass-through authentication.

Optional configuration property and the scenarios for use when using pass-through authentication

Property	Scenario in which a user can still bind by using pass-through authentication
temporarily-locked-due-to-failures	The account is locked temporarily because of too many failed attempts.
permanently-locked-due-to-failures	The account is locked permanently because of too many failed attempts.
locked-due-to-idle-interval	The account is locked because the user has not authenticated recently.

Property	Scenario in which a user can still bind by using pass-through authentication
locked-due-to-maximum-reset-age	The account is locked because an administrator recently reset the password, and the user failed to specify a new password within the allotted time frame.
password-is-expired	The password is expired.

Troubleshooting server performance issues

The following are the most common reasons that customers encounter performance issues and some suggestions for diagnosing and addressing these issues.

Slow password storage schemes

Slow password storage schemes are configurable schemes designed to use a lot of CPU and memory to thwart attackers, but they can affect legitimate operations like password validation and authentication.

The following password storage schemes are intentionally expensive:

- PBKDF2
- bcrypt
- scrypt
- Argon2
- The MD5, SHA-2-256 and SHA-2-512 variants of the crypt scheme

These schemes are designed to consume a significant amount of CPU, and memory in some cases, to increase the amount of resources an attacker must expend to crack a password if they happen to get access to the password's encoded representation. This same cost is also incurred for legitimate operations involving the password, including encoding clear-text passwords during account creation and password changes and when validating passwords during authentication. You can configure these schemes to adjust the amount of resources they consume, and you should configure them so that the resource consumption under expected peak load does not exceed the capacity of the topology.

Additionally, if you are initially populating the server using an LDIF import that contains clear-text passwords, using one of these schemes can cause the LDIF import to proceed at a small fraction of the rate that could be achieved with a faster storage scheme, such as one that uses a 256-bit or 512-bit salted SHA-2 digest. In such cases, you might import the data using a faster scheme and then change the configuration to make the desired scheme the new default, and mark the scheme used for import as deprecated. As a result, accounts with passwords encoded using the import scheme are automatically re-encoded with the new scheme the first time that the user successfully authenticates using that password.

Database size versus memory capacity

Best performance is achieved when the contents of the database can be fully cached in memory.

If possible, the amount of memory allocated to PingDirectory Server should be large enough so that all of the data can fit in the database cache. Memory sizing estimates appear at the end of the output when you run the `import-ldif` tool.

If the amount of data that needs to be stored exceeds the available memory capacity of the individual systems on which the server runs, there are a couple options:

- Break up the data into segments that are small enough to be fully cached on their own, and use PingDirectoryProxy Server's entry balancing functionality to make them appear as a single logical set.
- Tune the server so that the most frequently accessed information can be served from memory while disk access might be required for other data. In disk-bound deployments, you can use backend

configuration options such as cache mode, uncached attribute criteria, and uncached entry criteria to help prioritize what data should remain in memory versus what can be retrieved from disk.

Large number of access control rules
As the number of access control rules increases, so does the potential costs of determining whether a client is allowed to request a given operation and of paring down search result entries based on the data that the client is permitted to access.

The server might need to re-evaluate all access control rules after certain update operations, including modify DN operations, to determine whether these are affected by the change.

In many cases, deployments with an extremely large number of access control rules, especially those with large numbers of branches in which the same structure might be repeated across each of these branches, might be able to leverage parameterized access control instructions (ACIs) to dramatically reduce the number of access control rules that need to be defined and evaluated. In other cases, it is possible to refactor the access control configuration to achieve the same effect but with far fewer rules.

Large static groups
PingDirectory Server supports large static groups with hundreds of thousands of members or more, but these can have a significant impact on performance. Consider replacing large static groups with dynamic groups.

It can be expensive in terms of system resources to update the large static group to add or remove members because each update requires rewriting the entire entry. It can also be expensive to decode the entry when retrieving it from the database, even if it is held in the database cache, and to return the list of members to a client in search results.

If possible, consider replacing large static groups with dynamic groups. Dynamic groups automatically determine their membership based on criteria defined in LDAP URLs. A dynamic group consumes little memory and disk space and does not need to be altered as entries are created, updated, and removed. In some cases, it is possible to use virtual static groups in conjunction with dynamic groups to create entries that behave like static groups for read operations. This lets you maintain compatibility with applications that only understand static groups but use dynamic groups behind the scenes to determine membership.

If it is not possible to eliminate large static groups, you can enable the static group entry cache. While this does not reduce the performance impact of updating large static groups, it can make it much more efficient to access those groups for read operations.

Large index ID sets
Indexes store data that make it possible to quickly retrieve matching entries during a search. As the size of an ID set increases, so does the potential resource cost of accessing the ID set.

Each index record maps an index key to a list of the entry IDs for the entries that match that key. If you have an equality index for a given attribute, there is a key for each unique value for that attribute and the values for each key of the IDs of the entries that contain that value. For substring indexes, there can be multiple keys for the same attribute value (one for each unique six-character substring within the value), and the same substring can apply to many different values of the same attribute.

For an attribute index, you can store ID sets in one of two ways that each affect performance differently:

- In the regular (non-exploded) case, each index key occurs once, and the value of this key is the entire list of IDs for entries that match the key. The ID set for a non-exploded index key can be retrieved quickly because only a single database read is required. However, as the size of the ID set grows, the cost of updating it grows because it is necessary to replace the entire set, which requires larger amounts of disk I/O and can place an increased burden on the database cleaner.
- In the exploded case, the same key can be stored multiple times (one for each entry that matches that key), with each instance of the key associated with a different entry ID. Updating the ID set for an exploded key is always fast because the writes are small, but the cost of reading the ID set increases with the number of IDs.

You can also use composite indexes. These offer many advantages over attribute indexes and when they can be used. Some of these advantages, such as the ability to configure a base DN or combinations of attributes, do not have any effect on performance with regard to large ID sets. They use a hybrid of the exploded and non-exploded approaches to maintaining the ID set, such that a large set can be split into multiple pieces, but each piece can have up to 5000 IDs rather than just one. This means that retrieving a large ID set from a composite index can be thousands of times cheaper than retrieving the same ID set from an exploded attribute index. Updating a large ID set in a composite index should be cheaper in terms of systems resources than updating the same ID set in a non-exploded attribute index because the write is much smaller.

In environments with performance problems related to very large index ID sets, you might consider the following options as a way to help improve performance:

- Consider reducing the index entry limit for that index. The index entry limit specifies the maximum number of IDs that an ID set can have for an index key.
  - If a key matches more entries than this limit, the server stops maintaining the index for that key, and attempts to access it behave as if it were unindexed, while the index continues to be maintained for keys matching smaller numbers of entries. If you don't have searches that depend only on those keys, then this is an excellent way of eliminating the cost of maintaining large ID sets. It isn't logical to set the index entry limit to a value that is a large percentage of the total number of entries in the server. In such cases, there might not be a significant performance difference between indexed and unindexed search performance, but there would be no need to maintain the associated large ID sets.
- If there are cases in which you need a large index entry limit, then consider increasing the limit only for that index, rather than increasing the default limit for all indexes in the backend.
  - The `index-entry-limit` property in the backend applies to all indexes that don't specify their own limit, but each index also offers an `index-entry-limit` property that, if set, overrides the index entry limit configured for the backend. As such, if you need a higher index entry limit for a particular index, set a higher limit just for that one index instead of raising the default limit for all indexes in the backend.
- For attribute indexes with keys matching a large number of entries, consider converting it to a composite index when possible.
  - Composite indexes can completely replace equality and ordering attribute indexes, and they can support "starts with" substring searches, regardless of whether they have "contains" or "ends with" components. Composite indexes can't currently replace approximate match indexes or substring searches that don't have a "starts with" component.
- Consider eliminating any unnecessary substring indexes.
  - As previously noted, substring indexes are more likely to have large ID sets than equality, ordering, or approximate match indexes because substring keys are generally smaller and any given substring key can apply to multiple attribute values. It's also not commonly understood that equality attribute indexes and equality composite indexes are used for substring searches with a "starts with" component. As such, a substring index is not used for substring searches with a "starts with" unless there isn't an equality index for that attribute.
  - If a substring index is defined for an attribute that isn't targeted by substring searches, or that is only targeted by substring searches that contain a "starts with" component (regardless of whether it also includes "contains" or "ends with" components), then that substring index is not necessary and can be removed. You can use access logs to determine the types of searches that clients are performing, and the `summarize-access-log` and `search-logs` tools might help with that.
- If a substring index is needed for a given attribute, then consider increasing the substring length for that index.
  - By default, the server creates a separate substring key for each unique six-character substring in an attribute value, and there might be cases in which the same six-character substring appears in several different values. If that occurs and causes substring keys to have large ID sets, then increasing the substring length for that index reduces the number of values that might share the any given key and can reduce the number of IDs associated with that key.

- As a last resort, consider tuning the exploded index threshold for an index (the number of entries that an ID set needs to have for a given key before it will transition from a non-exploded set to an exploded one) based on the expected usage for that index.
  - If search performance is more important than update performance for an attribute with large ID sets, then raising the exploded index threshold helps keep the ID set stored as a monolithic block of IDs. On the other hand, if update performance is more important than search performance, then lowering the exploded index threshold might help.

Missing indexes
Maintain indexes to determine if an index is missing.

Maintaining unneeded indexes can have a detrimental effect on performance, but it can also be detrimental if a needed index is missing.

The best way to identify expensive searches processed in the server is to examine the access logs for search operations with a high `etime` (elapsed processing time) value. After you identify these search operations, you can filter out any of these operations that do not need to be fast. For example, you might have applications that generate reports by performing inefficient searches, such as searches to retrieve all entries, and it is usually more acceptable for those searches to be slow.

For any remaining searches that are slow or that should be faster, the best way to understand why the search is expensive is to issue a search with the same base DN, scope, and filter, but request only the `debugsearchindex` attribute.

> ⓘ **Note:**
>
> `debugsearchindex` is a special attribute that makes the server return debug information about the index processing that is being performed in the course of evaluating the search, and how long it took to complete each step of the evaluation.

From this output, you can see which indexes were used and which could not be used because there was either no applicable index or the index entry limit had been exceeded for the target key. You can see expensive accesses to exploded indexes and identify indexes you want to add, indexes that can benefit from being converted to composite indexes, or indexes where you might need to increase the index entry limit. Alternatively, you can determine a different way to perform the search so that it does not depend on components that are unindexed or that match a large number of entries.

Configuring the Directory Server for Oracle compatibility

Configure the Directory Server for Oracle compatibility.

About this task

For companies that are migrating from an Oracle server to the PingDirectory Server, the PingDirectory Server provides a **`dsconfig`** batch file, `sun-ds-compatibility.dsconfig`, which describes the various components that can be configured to make the server exhibit behavior closer to an Oracle configuration.

Administrators can use the `sun-ds-compatibility.dsconfig` batch file to apply the Directory Server's configuration with the necessary **`dsconfig`** commands by uncommenting the example commands listed in the file and then running the **`dsconfig`** command specifying the batch file.

> ⓘ **Note:**
>
> This batch file is not comprehensive and must be used together with the **`migrate-sun-ds-config`** tool, located in the `bin` folder, or `bat` folder for Windows systems, during the migration process. Both the tool and the batch file overlap in some areas but provide good initial migration support from the Oracle server to a Ping Identity server.

Another useful tool is the **migrate-ldap-schema** tool in the `bin` folder, or `bat` folder for Windows systems, which migrates schema information from an existing LDAP server onto this instance. All attribute type and object class definitions that are contained in the source LDAP server are added to the targeted instance or written to a schema file.

Steps

1. From the Directory Server server root directory, open the `sun-ds-compatibility.dsconfig` file in the `docs` folder.

   You can use a text editor to view the file.
2. Read the file completely.
3. Apply any changes to the file by removing the comment symbol at any **dsconfig** command example, and then applying the **dsconfig** command specifying the batch file.

```
$ bin/dsconfig --no-prompt --bindDN "cn=Directory Manager" \
  --bindPassword "password" --batch-file /path/to/dsconfig/file
```

4. Run the **migrate-ldap-schema** tool to move the schema definitions on the source server to the destination Ping Identity server.

```
$ bin/migrate-ldap-schema
```

5. Run the **migrate-sun-ds-config** tool to see what differences exist in the Ping Identity configuration versus the Oracle configuration.
6. On the PingDirectory Server, run the following command.

```
$ bin/migrate-sun-ds-config
```

7. Test the server instance for further settings that might not have been set with the batch file, the **migrate-ldap-schema** tool, or the **migrate-sun-ds-config** tool.
8. If you notice continued variances in your configuration, contact your authorized support provider.

Supporting unindexed search requests

By default, the Directory Server denies all unindexed search requests, except for those issued by the bind distinguished names (DNs) that have the `unindexed-search` privilege.

About this task

This default behavior keeps the server from tying up worker threads on time-consuming, unindexed searches. However, you can turn off the enforcement of the `unindexed-search` privilege to allow any client to perform an unindexed search.

Steps

1. Set the `disabled-privilege` global configuration property to `unindexed-search` as follows.

```
$ bin/dsconfig set-global-configuration-prop \
  --set disabled-privilege:unindexed-search
```

2. If you choose to allow unindexed searches, cap the maximum number of concurrent unindexed search requests using the `maximum-concurrent-unindexed-searches` global configuration property.

```
$ bin/dsconfig set-global-configuration-prop \
  --set maximum-concurrent-unindexed-searches:2
```

3. Limit unindexed search privileges for particular clients using the `allow-unindexedsearches` property of the client connection policy.

   For more information about configuring client connection policies, see *Client connection policy configuration*.

Syncing passwords to PingOne

Create a direct attribute mapping to sync passwords from PingDirectory Server to PingOne.

About this task

PingDirectory Server maps the userPassword attribute to thepassword attribute to sync passwords with PingOne.

Steps

- To create a direct attribute mapping, run the following.

```
dsconfig create-attribute-mapping \
--map-name PingDirectory_to_PingOne_User_Map \
--mapping-name password \
--type direct \
--set from-attribute:userPassword
```

  PingDataSync Server can synchronize passwords that have been encrypted by PingDirectory Server or a hashed version of the password, depending on how an administrator chooses to store passwords on PingDirectory Server.
- To sync passwords from a generic relational database management system (RDBMS), create a direct attribute mapping with the from-attribute being whichever attribute the RDBMS uses to store the password.

  In the following example, the RDBMS uses the `dbPassword` attribute to store the password.

```
dsconfig create-attribute-mapping \
--map-name Generic_RDBMS_to_PingOne_User_Map \
--mapping-name password \
--type direct \
--set from-attribute:dbPassword
```

  RDBMS passwords cannot be encrypted and should be hashed with a scheme that PingDirectory Server recognizes.

  PingDataSync Server cannot synchronize passwords between PingOne systems because PingDataSync Server cannot retrieve passwords from PingOne.

Configuring PingOne to use SSO for the PingData Administrative Console

Allow administrative users to single sign-on (SSO) to the PingData admin console from PingOne.

Before you begin

You need the following to complete this process:

- A configured PingData server. This server will host the the PingOne administration console console that is being configured for SSO.
- A PingOne for Customers account. For more information, see *Getting started with PingOne for Customers*.
- Access to the the PingOne administration console console. For more information, see *Using the beta version of My Ping*.

Steps

1. In the PingOne administration console, add a link to the PingOne solutions home page.
   a. In the the PingOne administration console admin console, click **Add Environment**.

      If you're adding PingDirectory Server or PingDataGovernance Server to an existing environment, click the name of an environment, click the **Plus** icon and click **Add** to add PingDirectory.
   b. To create an environment, on the **Create Environment** page, select from **Customers**, **Workforce**, or **Custom**.
   c. Select **PingDirectory** and **PingOne for Customers**.
   d. Click **Next**.
   e. Select `It's already been deployed`.
   f. In the **Enter Admin URL** field, enter `https://<hostname>:<port>/console/login`, replacing the bracketed variables with the PingData server's hostname and HTTP port.
   g. Click **Next**.
   h. In the **Environment Name** field, enter a name for this environment.
   i. Optional: In the **Description** field, enter a description for the environment.
   j. From the **Region** list, select your data center region.
   k. From the **License** list, select the license for this environment.
   l. Click **Finish**.
2. To configure the matching administrator accounts for PingOne and the PingData server, go to the PingOne dashboard for the environment that will be used with the PingData server and repeat the following steps for each PingOne user for whom you want to enable SSO.
   a. In the PingOne administration console, on your environment line, click the **PingOne** icon.
   b. In PingOne, go to **Identities**.
   c. On the line of the administrative user you want to configure, click the **Expand** icon.
   d. Run the following **`dsconfig`** command against the PingData server, replacing the bracketed fields with the values of the administrative user.

      ```
      dsconfig create-root-dn-user --user-name <Username> \
        --set first-name:<Given Name> \
        --set last-name:<Family Name>
      ```
3. Register the Administrative Console with PingOne.
   a. Go to *Add an application - Web application* and follow the instructions in the **Add an OIDC application** subsection.
   b. Enter the application properties as shown in the following table.

Property	Value
Application Name	PingData Administrative Console
Description	Application for the PingData Administrative Console
Redirect URLs	`https://<hostname>:<port>/console/oidc/cb`
Attribute Mapping	'Username' = 'sub'

> ⓘ **Note:**
>
> Fill in the bracketed values in **Redirect URLs** with your PingData server's hostname and HTTP port.

4. Edit the listed properties for the newly created application so that the properties have the values show in the following table, following the instructions in *Edit an application - OIDC* in the PingOne Administration Guide.

Property	Value
Response Type	Code
Grant Type	Authorization Code
Token Endpoint Authentication Method	Client Secret Basic

5. Record the values for the following application properties to use in later steps:
   - Issuer
   - Client ID
   - Client Secret
6. Create a copy of the `PingDirectory/config/sample-dsconfig-batch-files/enable-pingone-admin-console-sso.dsconfig` file, leaving the source file as-is.
7. Open the copy of the file and replace the bracketed values with the values from step 5.
8. Run the file using the following command.

```
dsconfig --batch-file \
    enable-pingone-admin-console-sso-copy.dsconfig \
    --no-prompt
```

9. Click the link to the PingData server from the PingOne solutions home page.
   A PingOne sign on page displays.
10. Sign on using the administrative user credentials.
    The Administrative console index page displays.

Configuring Soft Deletes

PingDirectory Server 3.2.4 and later supports a soft-delete feature.

This feature preserves a deleted entry's attribute and uniqueness characteristics so it can be undeleted or permanently removed at a later date.

About soft deletes

Soft deletes preserve a deleted entry's attribute and uniqueness characteristics so it can be undeleted or permanently removed at a later date.

The standard implementation of an LDAP server allows adding, renaming, modifying, searching, comparing, and deleting one or more entries. By specification, the `delete` operation permanently removes an entry and its attributes in a Directory Information Tree (DIT) but records the changes in access, and optionally, audit and change logs. The `delete` operation severs any associations such as references and group memberships. Meta attributes, such as operational attributes, which can be unique to an entry like `entryUUID`, are lost or different if the same entry is re-added to the Directory Server.

There are cases where a company might want to preserve their deleted entries to allow for possible undeletion at a later date. For example, a company might want to retain account and subscriber entries for their users who leave but later rejoin. Artifacts that a user creates such as account histories, web pages, notes, can be tracked and recovered while a user is deleted or when the user returns as an active customer. Soft deletes facilitate this use-case.

A delete request can result in a soft delete either by the client explicitly requesting a soft delete or by the request matching criteria defined in an active soft delete policy. The soft-deleted entries are renamed by prefixing an `entryUUID` operational attribute to the DN and adding an auxiliary object class to the entry, `ds-soft-delete-entry`, which saves the entry in a hidden state. All active references and group memberships are then removed. While in this hidden state, clients cannot access soft-deleted entries under normal operating conditions. Only clients with the `soft-delete-read` privilege can interact with soft-deleted entries.

To allow soft deletes, the Directory Server's attribute uniqueness function has been relaxed to allow for the co-existence of a soft-deleted entry and an active entry with identical naming attributes, such as `uid`. For example, if a user John Smith was soft deleted, but a different John Smith was added to the user accounts system, both entries can reside in the DIT without conflict. One entry would exist in a soft-deleted state and the other in an active state. The Directory Server extends this capability by allowing multiple users with the same DN, who would normally conflict if active, to reside in the soft-deleted state.

Soft-deleted entries can be restored with an `undelete` operation. The same uniqueness constraints that apply when adding a new user to the Directory Server are enforced when a soft-deleted entry is undeleted. In the previous example, John Smith was soft-deleted, but a different John Smith with the same `uid` as the original John Smith was later added to the system. If the original John Smith was undeleted from its soft-deleted state, it would result in a conflict with the active John Smith entry. Administrators must modify the DN of the soft-deleted entry to avoid such conflicts.

Administrators can permanently remove a soft-deleted entry by performing a regular `delete` operation on it. This operation, called a hard delete, permanently removes a soft-deleted entry from the server. You can also permanently remove a regular non-soft-deleted entry using a hard delete. This is useful when the server is configured with a soft-delete policy that would otherwise turn a regular delete request into a soft delete.

The Directory Server provides tool arguments that can use the soft delete request control, the hard delete request control, and other controls necessary to process these operations. Procedures to show how to use these options are presented later in this section.

For replicated topologies, when a participating directory server soft deletes an entry, it notifies the other replicas in the topology to soft delete the same entry on its respective machine. The changelog backend also records these entries by annotating them using an attribute that indicates its soft-deleted state. Modification and hard deletes of soft-deleted entries are not recorded by default in the changelog but can be enabled in the server. For maximum compatibility, make sure all servers in the replication cluster support soft deletes and have identical soft delete configurations.

General tips on soft deletes

Administrators should be aware of the following tips about soft deletes:

- The LDAP SDK and Server SDK both fully support soft-deletes.
- There is little performance difference between retrieving a regular entry and a soft-deleted entry. However, there might be a performance impact when a search operation has to match criteria, such as `uid=john.smith`, for both active entries and soft-deleted entries. For example, if there is one active `uid=john.smith` entry and two soft-deleted `uid=john.smith` entries, it might take the server more time to retrieve and try to match the criteria before it can return the results.
- The soft delete feature fully supports uncached attributes and uncached entries. For more information, see *Uncached Attributes and Entries*.
- Soft-deletion is allowed for leaf nodes only. Soft-deletion of any parent entry is not allowed. Likewise, soft-deleted entries that have soft-deleted sub-entries are not allowed.
- There are two available state options for soft-deletes:
  - Administrators can permanently delete a soft-deleted entry or undelete the entry.
  - Administrators cannot soft-delete an already soft-deleted entry, which returns an `UNWILLING_TO_PERFORM` result code.

- Soft-deleted users have no privileges. Soft-deleted users do not have the ability to bind to the Directory Server or have authentication access. They cannot change their passwords and cannot undelete themselves. Soft-deleted entries also cannot be used as an authorization identity using the proxied authorization or immediate client control. The soft-delete process does not destroy privilege assignment. If a soft-deleted entry is undeleted, the restored entry retains the same privileges it originally had before being soft deleted. One possible exception to this are privileges assigned by virtual attributes that no longer match the newly-undeleted entry. Those entries do not retain their original privileges.
- Soft-deleted entries might not be accessible from alternate access methods like SCIM.
- Soft-deleted entries can be modified but not renamed. Administrators can search for all soft-deleted entries and the original source entry attributes can be updated as long as the administrator has modify privileges and access to the `soft-delete-read` privilege. Any attempt to rename a soft-deleted entry using a `modify DN` operation results in an `UNWILLING_TO_PERFORM` result code.
- Replication has access to the LDAP operations with soft delete controls. These operations are transmitted, processed, and replayed as high-level requests, which are re-played on remote replicated servers. The replication conflict-resolution mechanism handles soft-deleted entries like any regular entries. For example, if a soft delete is executed independently on two servers and is then replicated, this results in a replication conflict. For maximum compatibility, all servers in a replication cluster should support soft deletes and have identical soft delete configuration.
- Soft-deletes are supported in transactions. The processing workflow uses the transactions mechanism and maintains the context information necessary to rollback failures to soft delete or undelete.
- There are no special configuration steps to configure soft deletes on the Directory Proxy Server. The soft-deleted entry is routed directly to the underlying Directory Server. There is one exception: in an entry-balancing deployment, the Directory Proxy Server is responsible for routing the soft-deleted entry to the Directory Server containing the originally soft-deleted item. As with standard entry-balanced deployments, it is not possible to use `moddn` to undelete an entry to a different Directory Server.
- The default behavior includes soft-deleted entries as part of the **`export-ldif`** operation. If soft-deleted entries are to be excluded from export, administrators can use the `--excludeSoftDeleteEntries` option to filter out the entries.
- The soft delete feature can be used with users who have proxied authorization privileges.
- For customers using the PingDataSync Server, soft-deleted entries are not synchronized by the server. Modifications or deletes of a soft-deleted entry are ignored by the Data Sync Server, and do not appear in the changelog by default. An actual soft delete operation appears to the changelog as a regular `delete` operation, and an actual undelete operation appears in the changelog as a regular `add` operation.
- References to a deleted DN are not restored by the referential integrity plugin when you undelete a soft-deleted entry. For example, if you have referential integrity enabled and you soft-delete a DN that is a member of a static group, the referential integrity plugin removes this DN from the group's list of members. When you undelete the soft-deleted entry, the plugin does not add the entry back to the group.
- The soft delete policy configuration supports two new properties, `soft-delete-retention-time` and `soft-delete-retain-number-of-entries`, that perform the purging of soft-deleted entries. For more information, see *Configuring Soft-Delete Automatic Purging*.
- The root user account, such as `cn=Directory Manager`, has access to all of the controls needed to run the soft delete operations by default. For non-root users, you must grant access to these soft delete controls using access control rules. An example is shown in step 1 of *To Configure Soft Deletes as a Global Configuration*. The following soft delete controls are available to non-root users:

  Soft delete request control

  Allows the user to perform a soft delete operation. The object identifier (OID) for the control is 1.3.6.1.4.1.30221.2.5.20.

  Soft delete response control

Allows the server to hold the DN of the soft-deleted entry that results from a soft delete request. The OID for the control is 1.3.6.1.4.1.30221.2.5.21.

Hard delete request control

Allows the user to run a hard delete operation on the entry, regardless if it is a regular or soft-deleted entry. The OID for the control is 1.3.6.1.4.1.30221.2.5.22.

Undelete request control

Allows the user to undelete a soft-deleted entry using an ADD request. The OID for this control is 1.3.6.1.4.1.30221.2.5.23.

Soft-deleted entry access request control

Allows the user to search for any soft-deleted entries. The OID for this control is 1.3.6.1.4.1.30221.2.5.24.

> ⓘ **Note:**
>
> A bind DN with the `stream-values` privilege can perform operations that can reveal soft-deleted entries, even if that bind DN does not have permission to use the soft-deleted entry access request control. For example, if a user can successfully run **`dump-dns`** or **`ldap-diff`**, then that user can get a list of soft-deleted entry DNs or soft-deleted entry contents through the output of one of those tools.

Configuring soft deletes on the Server

Use any of the following methods to configure soft deletes on the Directory Server.

Method	Description
Using a soft-delete policy and global configuration property	Configure soft deletes by creating a soft-delete policy and a global configuration property. The soft-delete policy enables the feature on the server, while the global configuration property sets the controls used for the soft-delete requests. To enter a soft delete request, the **`ldapmodify`** or **`ldapdelete`** command requires the `--useSoftDelete` option. A delete request that does not have the `--useSoftDelete` option is treated as a hard delete, which permanently removes the entry.
Using connection criteria	Configure soft deletes by defining connection criteria within a client connection policy. Any clients that meet the criteria have their deletes processed as soft deletes.
Using request criteria	Configure soft deletes by defining request criteria within a client connection policy. Any client requests that meet the criteria have their deletes processed as soft deletes. ⓘ **Note:** You can define both connection criteria and request criteria. Both criteria can exist within a soft-delete

Method	Description
	policy. In this case, the connection criteria is processed first, then the request criteria.

Configuring soft deletes as a global configuration
Configure the soft delete feature by creating a soft delete policy and then setting the configuration property on the server. The presence of the soft-delete policy enables the feature on the server and allows the global configuration property to send the necessary soft delete requests.

About this task

For this configuration, use the `--useSoftDelete` option used with the **`ldapmodify`** or **`ldapdelete`** commands to send the delete using the soft delete request control. Without the `--useSoftDelete` option, any delete is processed as a hard delete.

To configure soft deletes as a global configuration:

Steps

1. Configure a soft delete policy using the **`dsconfig`** command.

   The soft delete configuration requires a soft delete policy, which enables the feature on the server.

   ```
   $ bin/dsconfig create-soft-delete-policy \
     --policy-name default-soft-delete-policy
   ```

2. Configure the soft delete as a global configuration property using the **`dsconfig`** command.

   This command sets up the soft delete controls necessary to send them as a request.

   ```
   $ bin/dsconfig set-global-configuration-prop \
     --set soft-delete-policy:default-soft-delete-policy
   ```

Configuring a user to use soft or hard delete controls
To use soft deletes, a user must have access to the appropriate controls. By default, only the Directory Manager has access to these controls.

About this task

The user must also have the **`soft-delete-read`** privilege. Access control instructions (ACIs) allow the user to:

- Modify target entries
- Use the soft delete and undelete controls
- Use the soft-deleted entry access control to modify soft-deleted entries
- Use the hard delete request control to permanently delete an soft-deleted entry

The **`uid=admin,dc=example,dc=com`** user that is installed with the sample data during setup already has an ACI giving it access to user entries as follows.

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

Steps

1. To restrict the scope, add the following ACIs to the base suffix or other point in the directory information tree (DIT) as required.

```
(targetcontrol="1.3.6.1.4.1.30221.2.5.20||1.3.6.1.4.1.30221.2.5.21")
(version 3.0; acl "Allow admins to use the Soft Delete Request Control and
 Soft Delete Response Control";
allow (read) userdn="ldap:///uid=admin,dc=example,dc=com";)

(targetcontrol="1.3.6.1.4.1.30221.2.5.22") (version 3.0; acl "Allow admins
 to use the Hard Delete
Request Control";allow (read) userdn="ldap:///uid=admin,dc=example,dc=com";)

(targetcontrol="1.3.6.1.4.1.30221.2.5.23") (version 3.0; acl "Allow admins
 to use the Undelete
Request Control";allow (read) userdn="ldap:///uid=admin,dc=example,dc=com";)


(targetcontrol="1.3.6.1.4.1.30221.2.5.24") (version 3.0; acl "Allow admins
 to use the Soft-Deleted
Entry Access RequestControl"; allow (read) userdn="ldap:///
uid=admin,dc=example,dc=com”;)
```

2. Add the **`ds-privilege-name`** attribute to the user with the value **`soft-delete-read`**.

```
$ ./bin/ldapmodify -s -p 1389 -D uid=admin,dc=example,dc=com -w password
# Successfully connected to localhost:1389.

dn: uid=user.10,ou=people,dc=example,dc=com
changetype: delete

# Deleting entry uid=user.10,ou=people,dc=example,dc=com ...
# Result Code:  0 (success)
# Soft Delete Response Control:
#      OID:  1.3.6.1.4.1.30221.2.5.21
#      Soft-Deleted Entry DN:  entryUUID=8dbe8cb4-1aa3-41c5-88ec-
a6280eeff918+uid=user.10,ou=People,dc=example,dc=com
```

Searching for soft deletes

Soft-deleted entries are excluded from normal LDAP searches because they represent deleted entries. The updated **`ldapsearch`** tool supports these types of searches.

About this task

There are three different ways to search for soft-deleted entries.

Steps

- To perform a base-level search on a soft-deleted entry by distinguished name (DN), run the **`ldapsearch`** command and specify the base DN of the specific soft-deleted entry that you are searching for.
- To filter your search by `ds-soft-delete-entry` object class, run a search for all soft-deleted entries with the **`ldapsearch`** command with a filter on the `ds-soft-delete-entry` object class.

- To return soft-deleted entries, use the soft-delete-entry-access-control with the LDAP search.

  The **ldapsearch** tool provides a shortcut option, `--includeSoftDeletedEntries`, that sends the control to the server for processing. The control allows for the following search possibilities:

  - Return only soft-deleted entries.
  - Return non-deleted entries along with soft-deleted entries.
  - Return only soft-deleted entries in undeleted form.

Running a base-level search on a soft-deleted entry
Use the command line to run a base-level search on a soft-deleted entry.

Steps

- Run the **ldapsearch** command using the base distinguished name (DN) of the specified soft-deleted entry.

```
$ bin/ldapsearch \
  --baseDN entryUUID=4e9b7847-edcb-3791-
b11b-7505f4a55af4+uid=user.1,ou=People,dc=example,dc=com \
  --searchScope base "(objectClass=*)"
```

```
# Soft-deleted entry DN:
# entryUUID=4e9b7847-edcb-3791-
b11b-7505f4a55af4+uid=user.1,ou=People,dc=example,dc=com
dn: entryUUID=4e9b7847-edcb-3791-
b11b-7505f4a55af4+uid=user.1,ou=People,dc=example,dc=com
objectClass: top
objectClass: person
objectClass: organizationalPerson
objectClass: inetOrgPerson
objectClass: ds-soft-delete-entry
postalAddress: Aartjan Aalders$59748 Willow Street$Green Bay, TN 66239
postalCode: 66239
description: This is the description for Aartjan Aalders.
uid: user.1
userPassword: {SSHA}RdBCwQ2kIw57LukRthjrFBS/oFylJARnmTnorA==
employeeNumber: 1
initials: AKA
givenName: Aartjan
pager: +1 197 025 3730
mobile: +1 890 430 9077
cn: Aartjan Aalders
sn: Aalders
telephoneNumber: +1 094 100 7524
street: 59748 Willow Street
homePhone: +1 332 432 4295
l: Green Bay
mail: user.3@maildomain.net
st: TN
```

Running a filtered search by soft-delete-entry object class
Retrieve all soft-deleted entries using the `ds-soft-delete-entry` object class.

Steps

- Run the **ldapsearch** command to retrieve all soft-deleted entries using the `ds-soft-delete-entry` object class.

```
$ bin/ldapsearch --baseDN dc=example,dc=com \
```

```
  "(objectclass=ds-soft-delete-entry)"
```

Running a search using the soft delete entry access control

The following examples use the `--includeSoftDeleteEntries {with-non-deleted-entries | without-non-deleted-entries | deleted-entries-in-undeleted-form}` option, which uses the soft delete entry access control.

About this task

You can use the `--control` option with the soft delete entry access control symbolic name, `softdeleteentryaccess`, or the `--control` option with the actual soft delete entry access control OID, `1.3.6.1.4.1.30221.2.5.24`.

Steps

1. To return only soft-deleted entries, run **`ldapsearch`** using the `--includeSoftDeletedEntries` option with the value of `without-non-deleted-entries`.

   ```
   $ bin/ldapsearch --baseDN dc=example,dc=com \
     --includeSoftDeletedEntries without-non-deleted-entries \
     --searchScope sub "(objectclass=*)"
   ```

2. To return non-deleted entries along with soft-deleted entries, run **`ldapsearch`** using the `--includeSoftDeletedEntries` option with the value of `with-non-deleted-entries`.

   ```
   $ bin/ldapsearch --baseDN dc=example,dc=com \
     --includeSoftDeletedEntries with-non-deleted-entries \
     --searchScope sub "(objectclass=*)"
   ```

3. To return only soft-deleted entries in undeleted form, run **`ldapsearch`** using the `--includeSoftDeletedEntries` option with the value of `deleted-entries-in-undeleted-form`.

   Some applications require access to all entries in the server, including both active and soft-deleted entries.

   The following command returns all entries that were soft-deleted but presents it in a form that is similar to a regular entry with the soft-delete DN in comments. This regular entry format does not show the actual soft-deleted DN but displays it in an "undeleted" form even though it is not actually "undeleted". The object class, `ds-soft-delete-entry`, is also not displayed.

   ```
   $ bin/ldapsearch --baseDN dc=example,dc=com \
     --includeSoftDeletedEntries deleted-entries-in-undeleted-form \
     --searchScope sub "(ds-soft-delete-from-dn=*)"

   # Soft-deleted entry DN:
   # entryUUID=2b5511e2-7616-389b-ab0c-025c805ad32c
   +uid=user.14,ou=People,dc=exam-
   ple,dc=com
   dn: uid=user.14,ou=People,dc=example,dc=com
   objectClass: top
   objectClass: person
   objectClass: organizationalPerson
   objectClass: inetOrgPerson
   postalAddress: Abdalla Abdou$78929 Hillcrest Street$Elmira, ME 93080
   postalCode: 93080
   description: This is the description for Abdalla Abdou.
   uid: user.14
   userPassword: {SSHA}7GkzWiMiU12m5m+xBV+ZsoX3gVacMcRtSwDTFg==
   employeeNumber: 14
   initials: AFA
   givenName: Abdalla
   ```

```
pager: +1 307 591 4870
mobile: +1 401 069 1289
cn: Abdalla Abdou
sn: Abdou
telephoneNumber: +1 030 505 6190
street: 78929 Hillcrest Street
homePhone: +1 119 487 2328
l: Elmira
mail: user.14@maildomain.net
st: ME
```

Undeleting a soft-deleted entry using the same RDN

When you decide to undelete a soft-deleted entry, if the original Relative Distinguished Name (RDN), such as `uid=user.1`, is still available, you can perform the undelete using that same RDN.

About this task

To undelete a soft-deleted entry, use **`ldapmodify`** with the `--allowUndelete` option and target the specific soft-deleted entry that you want to restore. In an LDIF file or from the command line, specify the `dn:<target entry>` attribute, which is the distinguished name (DN) that the entry is undeleted to, and the `ds-undelete-from-dn` attribute, which is the entry that is undeleted from. An undelete requires the `add` changetype so that the entry can be re-added to the server.

Steps

- To undelete a soft-deleted entry using the same RDN, run the command **`ldapmodify`** with the `--allowUndelete` option and target the specific soft-deleted entry that you want to restore.

  The first DN is the entry to undelete to and the `ds-undelete-from-dn` is the soft-delete entry to undelete from.

  ```
  $ bin/ldapmodify --allowUndelete
  dn: uid=user.1,ou=People,dc=example,dc=com
  changetype:add
  ds-undelete-from-dn: entryUUID=4e9b7847-edcb-3791-b11b-
  7505f4a55af4+uid=user.1,ou=People,dc=example,dc=com
  ```

  The `--allowUndelete` option sends the soft undelete request control to the server.

Undeleting a soft-deleted entry using a new RDN

In some cases, you can allocate the original Relative Distinguished Name (RDN), `uid=user.1`, to a new user. This is permitted when the entry is in a soft-deleted state. To properly undelete this entry, you must specify a new RDN value you can use to restore the entry.

About this task

In this case, specifying the RDN of `uid=user.5` undeletes the original entry, but with the new distinguished name (DN) in the following example, and the `uid` attribute on the entry is updated with the new value of `user.5`. All other attributes of the users entry, including the `entryUUID`, remain unchanged.

To undelete a soft-deleted-entry using a new RDN:

Steps

1. Run the command **`ldapmodify`** to undelete a soft-deleted entry that has an original RDN, `uid=user.1`, to a new RDN, `uid=user.5`.

   > ⓘ **Note:**
   >
   > If you specify a DN that already exists in the Directory Server as a normal entry, this leads to an `entry already exists` error. Ensure the DN that you are undeleting the entry to does not already exist.

   ```
   $ bin/ldapmodify --allowUndelete
   dn: uid=user.5,ou=People,dc=example,dc=com
   changetype:add
   ds-undelete-from-dn: entryUUID=4e9b7847-edcb-3791-
   b11b-7505f4a55af4+uid=user.1,ou=People,dc=example,dc=com
   ```

2. To view the results, run **`ldapsearch`**.

   ```
   dn: uid=user.5,ou=People,dc=example,dc=com
   objectClass: top
   objectClass: person
   objectClass: organizationalPerson
   objectClass: inetOrgPerson
   postalAddress: Aartjan Aalders$59748 Willow Street$Green Bay, TN 66239
   postalCode: 66239
   description: This is the description for Aartjan Aalders.
   uid: user.5
   userPassword: {SSHA}RdBCwQ2kIw57LukRthjrFBS/oFylJARnmTnorA==
   employeeNumber: 1
   initials: AKA
   givenName: Aartjan
   pager: +1 197 025 3730
   mobile: +1 890 430 9077
   cn: Aartjan Aalders
   sn: Aalders
   telephoneNumber: +1 094 100 7524
   street: 59748 Willow Street
   homePhone: +1 332 432 4295
   l: Green Bay
   mail: user.3@maildomain.net
   st: TN
   entryUUID=4e9b7847-edcb-3791-b11b-7505f4a55af4
   ```

   The RDN and the `uid` attribute has changed.

Modifying a soft-deleted entry

Modify a soft-deleted entry the same as a regular entry using the **`ldapmodify`** tool. The entry remains hidden in its soft-deleted state after the change.

About this task

Soft-deleted entries can be modified like any regular entry. The only restriction is that you cannot change the distinguished name (DN) or run a `moddn` operation. To move a soft-deleted entry from one machine to another, use the **`move-subtree`** command and specify the DN of the soft-deleted entry. For more information, see *To Move an Entry from One Machine to Another*.

> ⓘ **Note:**

To modify a soft-deleted entry, the user needs the `soft-delete-read` privilege to access the soft-deleted entry.

Steps

- To modify a soft-deleted entry, run the **`ldapmodify`** command and specify the soft-deleted DN.

```
$ bin/ldapmodify
dn: entryUUID=4e9b7847-edcb-3791-
b11b-7505f4a55af4+uid=user.1,ou=People,dc=example,dc=com
changetype:modify
replace:telephoneNumber
telephoneNumber: +1 390 103 6918
# Processing MODIFY request for entryUUID=4e9b7847-edcb-3791-
b11b-7505f4a55af4+uid=user.1,ou=People,dc=example,dc=com
# MODIFY operation successful for DN entryUUID=4e9b7847-edcb-3791-
b11b-7505f4a55af4+uid=user.1,ou=People,dc=example,dc=com
```

Hard deleting a soft-deleted entry

Use this section for instructions on hard deleting a soft-deleted entry from the server when soft-deleted entries are configured as global configuration for requests or configured using a connection or request criteria.

About this task

Consider the following when hard deleting a soft-deleted entry:

Steps

- To permanently remove a soft-deleted entry from the server, run **`ldapdelete`** on the soft-deleted entry for soft-deleted entries.
- To hard delete a soft-deleted entry, use **`ldapdelete`** with the `--useHardDelete` option.

  The Hard Delete Request Control works with soft deletes. It applies when soft delete policies are in place as a means to override soft deletes requests. If soft deletes are configured, running **`ldapdelete`** with the Hard Delete Request Control, such as using the `--useHardDelete` option, guarantees that any entry permanently deletes.

Hard deleting a soft-deleted entry (global configuration)

About this task
Permanently remove a soft-deleted entry from the Directory Server.

Steps

- To permanently remove a soft-deleted entry, run **`ldapdelete`** on the soft-deleted entry.

  The following example assumes that you configured soft deletes as a global configuration for requests.

```
$ bin/ldapdelete \
  entryUUID=4e9b7847-edcb-3791-
b11b-7505f4a55af4+uid=user.1,ou=People,dc=example,dc=com

Processing DELETE request for entryUUID=4e9b7847-edcb-3791-b11b-
7505f4a55af4+uid=user.1,ou=People,dc=example,dc=com
DELETE operation successful for DN entryUUID=4e9b7847-edcb-3791-b11b-
```

```
7505f4a55af4+uid=user.1,ou=People,dc=example,dc=com
```

> **Note:**
>
> You cannot soft-delete an already soft-deleted entry. If you use the **`--useSoftDelete`** subcommand with the **`ldapdelete`** operation on a soft-deleted entry, an error message generates.
>
> ```
> DELETE operation failed. Result Code: 53 (Unwilling to Perform) Diagnostic
> Message: DELETE operation failed.
> ```

Hard deleting a soft-deleted entry (connection or request criteria)

About this task
Permanently remove a soft-deleted entry from the Directory Server.

Steps

- To permanently remove a soft-deleted entry run **`ldapdelete`** with the **`--useHardDelete`** subcommand on the soft-deleted entry.

  The following example assumes that you configured soft deletes using a connection or request criteria.

  ```
  $ bin/ldapdelete --useHardDelete \
    entryUUID=4e9b7847-edcb-3791-
  b11b-7505f4a55af4+uid=user.1,ou=People,dc=example,dc=com
  ```

Configuring soft deletes by connection criteria

Use this section for instructions on enabling and disabling soft deletes with connection criteria.

The Directory Server supports soft deletes where any delete operation is treated as a soft-delete request as long as the LDAP client meets the connection criteria.

To configure soft deletes:

- Define the connection criteria used in a client connection policy.
- Configure the soft delete connection criteria in the soft-delete policy.

Enabling soft deletes by connection criteria

Before you begin
Configure a soft-delete policy and global configuration, as shown in *Configuring Soft Deletes as a Global Configuration*.

Steps

1. Create a connection criteria using **`dsconfig`** and name it `Internal Applications`.

   In the following example the soft delete connection criteria is configured for a member of a line of business (LOB) applications group connecting from the 10.8.1.0 network.

   ```
   $ bin/dsconfig create-connection-criteria \
     --criteria-name "Internal Applications" \
     --type simple \
     --set included-client-address:10.8.1.0/8 \
     --set "all-included-user-group-dn:cn=LOB
    Applications,ou=Groups,dc=example,dc=com"
   ```

2. Set the `auto-soft-delete-connection-criteria` property to the soft-delete connection criteria you created in step 1.

```
$ bin/dsconfig set-soft-delete-policy-prop \
  --policy-name default-soft-delete-policy \
  --set "auto-soft-delete-connection-criteria:Internal Applications"
```

Disabling soft deletes by connection criteria

About this task
Disable soft deletes by connection criteria.

Steps

- Reset the `auto-soft-delete-connection-criteria` property on the soft-delete policy.

```
$ bin/dsconfig set-soft-delete-policy-prop \
  --policy-name default-soft-delete-policy \
  --reset auto-soft-delete-connection-criteria
```

Configuring soft deletes by request criteria

Soft deletes can be configured using request criteria within a client connection policy. All delete requests that meet the request criteria is treated as a soft delete. The presence of a soft delete by connection criteria is exclusive of the soft delete by request criteria. Both can be present in a soft-delete policy.

Enabling soft deletes by request criteria

Steps

1. Configure a soft-delete policy and global configuration as shown in “Configuring Soft Deletes as a Global Configuration”.
2. Configure request criteria for soft deletes. The soft delete request criteria is configured for an external delete request from a member of the Internal Applications group matching an entry with object class "inetorgperson" with the request excluding the Soft Delete Request Control and the Hard Delete Request Control.

```
$ bin/dsconfig create-request-criteria \
  --criteria-name "Soft Deletes" \
  --type simple \
  --set "description:Requests for soft delete" \
  --set operation-type:delete \
  --set operation-origin:external-request \
  --set "connection-criteria:Internal Applications" \
  --set not-all-included-request-control:1.3.6.1.4.1.30221.2.5.20 \
  --set "all-included-target-entry-filter:(objectClass=inetorgperson)"
```

3. In the soft delete policy created in step 1, set the `auto-soft-delete-connection-criteria` property to the simple criteria created in the previous step.

```
$ bin/dsconfig create-soft-delete-policy \
  --policy-name default-soft-delete-policy \
  --set "auto-soft-delete-request-criteria:Soft Deletes"
```

Disabling soft deletes by request criteria

Steps

- To disable soft deletes by request criteria, reset the soft-delete policy.

```
$ bin/dsconfig set-soft-delete-policy-prop \
  --policy-name default-soft-delete-policy \
  --reset auto-soft-delete-request-criteria
```

Configuring soft delete automatic purging

By default, the Directory Server retains soft-deleted entries indefinitely. For companies that want to set up automatic purging of soft-deleted entries, the server provides two properties on the Soft Delete Policy that can be configured for either the maximum retention time for all soft-deleted entries and/or the retained number of soft-deleted entries. These changes take effect without requiring a server restart.

Configuring soft-delete automatic purging

About this task

You can change either the retention time or the retained number of entries to enable automatic purging. By default, both are set to an indefinite retention time and number of entries. The time unit of milliseconds (ms), seconds (s), minutes (m), hours (h), days (d), or weeks (w), may be preceded by an integer to specify a quantity for that unit, such as "1 d", "52 w", etc. Once you configure the properties, the changes take effect immediately without the need for a server restart.

Note that the server will delete all of the soft-deleted entries according to the policy in effect. If the policy is changed while entries are in the process of being deleted, the new policy takes effect after the in-process batch of entries is deleted and applies to any remaining soft-deleted entries going forward according to the new policy.

Steps

1. Retrieve the name of the Soft Delete Policy in effect using the **`dsconfig`** command. For this example, the Soft Delete Policy is called `default-soft-delete-policy`.

```
$ bin/dsconfig get-global-configuration-prop \
  --property soft-delete-policy
```

2. Do one or both of the following:
   - Run **`dsconfig`** to set the retention time for soft-deleted entries.

```
$ bin/dsconfig set-soft-delete-policy-prop \
  --policy-name default-soft-delete-policy \
  --set "soft-delete-retention-time:52 w"
```

   - Run **`dsconfig`** to set the retained number of soft-deleted entries.

```
$ bin/dsconfig set-soft-delete-policy-prop \
  --policy-name default-soft-delete-policy \
  --set soft-delete-retain-number-of-entries:1000000
```

3. The Soft Delete Policy must be assigned to the global configuration if it has not been assigned yet.

```
$ bin/dsconfig set-global-configuration-prop \
  --set soft-delete-policy:default-soft-delete-policy
```

Disabling soft-delete automatic purging

About this task

You can disable Soft-Delete automatic purging using the **dsconfig** command. The change takes effect immediately without the need of a server restart. However, if the server is in the middle of an automatic soft-delete purging, it may continue to purge entries until the next time it evaluates the Soft Delete Policy.

Steps

- Run `dsconfig` to reset the Soft-Delete Policy properties that control automatic purging: `soft-delete-retention-time` and `soft-delete-retain-number-of-entries`.

```
$ bin/dsconfig set-soft-delete-policy-prop \
  --policy-name default-soft-delete-policy \
  --reset soft-delete-retention-time \
  --reset soft-delete-retain-number-of-entries
```

Soft and hard delete processes

See the following tables for the results of actions taken during the soft and hard delete processes.

If no automatic soft delete criteria is configured

The following table summarizes the resulting actions of a `DELETE` operation for soft deletes.

Action	Result
Delete	Performs a hard delete on the entry.
Delete with the Soft Delete Request Control	Performs a soft delete on the entry.
Delete of a soft-deleted entry	Performs a hard delete on the entry.
Delete of a soft-deleted entry with the Soft Delete Request Control	Not allowed. Generates an `UNWILLING_TO_PERFORM` error.
Delete with a Hard Delete Request Control	Performs a hard delete on the entry.
Delete of a soft-deleted entry with the Hard Delete Request Control.	Performs a hard delete on the entry.

If soft delete connection or request criteria is configured

The following table summarizes the resulting actions of a `DELETE` operation for soft deletes configured by connection criteria or request criteria.

Action	Result
Delete not matching criteria	Performs a hard delete on the entry.
Delete matching criteria	Performs a soft delete on the entry.

Action	Result
Delete of a soft-deleted entry not matching criteria	Performs a hard delete on the entry.
Delete of a soft-deleted entry matching criteria	Performs a hard delete on the entry.
Delete not matching criteria with the Hard Delete Request Control	Performs a hard delete on the entry.
Delete matching criteria with the Hard Delete Request Control	Performs a hard delete on the entry.
Delete with Soft and Hard Delete Request Controls	Not allowed. Generates an `UNWILLING_TO_PERFORM` error.

Soft delete controls and tool options

See the following tables for summaries of soft delete controls and tools.

Soft delete OIDs

The following table shows the OpenIDs (OIDs) for each soft delete control. The soft delete OIDs are defined in the LDAP SDK generated API documentation.

OID Type	OID
Soft Delete Request Control	1.3.6.1.4.1.30221.2.5.20
Soft Delete Response Control	1.3.6.1.4.1.30221.2.5.21
Hard Delete Request Control	1.3.6.1.4.1.30221.2.5.22
Soft Undelete Request Control	1.3.6.1.4.1.30221.2.5.23
Soft Delete Entry Access Control	1.3.6.1.4.1.30221.2.5.24

Soft delete tool options

The following table shows the new tool options available for the soft delete operations.

Operation	Options
`ldapdelete` / `ldapmodify`	`--useSoftDelete/-s`. Process `DELETE` operations with the Soft Delete Request Control, whereby entries are renamed and hidden instead of being permanently deleted. The Directory Server must be configured to allow soft deletes. ⓘ **Note:** Any entries in the LDIF file with the changetype of `delete` are processed as a soft-delete request.
`ldapdelet`	`--useHardDelete`. Process `DELETE` operations with the Hard Delete Request Control, which bypasses any soft delete policies and processes the delete request immediately without retaining the entry as a soft-deleted entry. The Directory Server must be configured to allow soft deletes.
`ldapsearch`	`--includeSoftDeletedEntries {with-non-deleted-entries | without-non-deleted-entries | deleted-entries-in-undeleted-form}`. Process search operations with the soft delete entry access control. Soft delete search options are as follows: **with-non-deleted-entries** Returns all entries matching the search criteria with the results, including non-deleted and soft-deleted entries. **without-non-deleted-entries** Returns only soft-deleted entries matching the search criteria. **deleted-entries-in-undeleted-form** Returns only soft-deleted entries matching the search criteria with the results returned in their undeleted entry form. Users must have access to the Soft Delete Entry Access Control to search for soft-deleted entries.
`ldapmodify`	`--allowUndelete`. Process `ADD` operations, which include the `ds-undelete-from-dn` attribute as undelete requests. Undelete requests re-add previously soft-deleted entries back to the server as non-deleted entries by providing the Undelete Request Control with the `ADD` operation. The Directory Server must be configured to allow soft deletes to process any undelete requests and the client user must have the `soft-delete-read` privilege.

Soft delete OID symbolic names using with the --control/-J option

The following table shows the symbolic names that can be used with the server's LDAP commands using the `--control/-J` option.

Control	Symbolic Name
Soft Delete Request Control	softdelete

Control	Symbolic Name
Hard Delete Request Control	harddelete
Soft Undelete Request Control	undelete
Soft Delete Entry Access Control	softdeleteentryaccess

Monitoring soft deletes

The Directory Server provides monitoring entries and logs to track all soft delete operations. The access and debug logs do not have any options specific for soft deletes.

New monitor entries
Two new monitor entries are present for a backend monitor entry.

Administrators see the following additional monitor entries on `cn=userRoot Backend,cn=monitor`:

`ds-soft-delete-entry-operations-count`

Displays the number of soft deletes performed on the backend since server startup.

`ds-undelete-operations-count`

Displays the number of undeletes performed on the backend since server startup.

`ds-backend-soft-deleted-entry-count`

Displays the current number of soft-deleted entries in the database.

`ds-auto-purged-soft-deleted-entry-count`

Displays the current number of soft-deleted entries purged since the backend or server was restarted.

Monitoring soft deletes
Monitor soft deletes using the **`ldapsearch`** command.

Steps

- Run **`ldapsearch`** on the `cn=userRoot Backend,cn=monitor` branch using a search criteria targeting the `ds-backend-monitor-entry` object class.

```
$ bin/ldapsearch --baseDN "cn=userRoot Backend,cn=monitor" \
  --searchScope sub "(objectclass=ds-backend-monitor-entry)"
```

```
dn: cn=userRoot Backend,cn=monitor
objectClass: top
objectClass: ds-monitor-entry
objectClass: ds-backend-monitor-entry
objectClass: extensibleObject
cn: userRoot Backend
ds-backend-id: userRoot
ds-backend-base-dn: dc=example,dc=com
ds-backend-is-private: FALSE
ds-backend-entry-count: 200001
ds-backend-soft-deleted-entry-count: 1000
ds-soft-delete-operations-count: 40
ds-undelete-operations-count: 20
```

```
ds-auto-purged-soft-deleted-entry-count: 0
ds-base-dn-entry-count: 200001 dc=example,dc=com
ds-backend-writability-mode: enabled
```

Access logs

The access log records the LDAP operations corresponding to soft delete and undelete for `DELETE`, `SEARCH`, `MODIFY`, and `ADD` operations with the related soft-deleted values.

The access log does not require any configuration for soft delete.

`DELETE` (soft-delete) operations

The access log displays the following.

```
[14/May/2012:09:40:16.942 -0500] DELETE RESULT conn=18 op=1 msgID=2
dn="uid=user.1,ou=People,dc=example,dc=com" resultCode=0 etime=30.367
softDeleteEntryDN="entryUUID=4e9b7847-edcb-3791-
b11b-7505f4a55af4+uid=user.1,
ou=People,dc=example,dc=com"
```

`SEARCH` operations for soft-deleted entries

The access log displays the following.

```
[14/May/2012:09:40:52.320 -0500] SEARCH RESULT conn=19 op=1 msgID=2
base="dc=example,dc=com" scope=2 filter="(objectclass=ds-soft-delete-
entry)"
attrs="ALL" resultCode=0 etime=1.631 entriesReturned=1
```

`MODIFY` operations of soft-deleted entries

The access log displays the following.

```
[14/May/2012:09:42:43.679 -0500] MODIFY RESULT conn=20 op=1 msgID=1
dn="entryUUID=4e9b7847-edcb-3791-
b11b-7505f4a55af4+uid=user.1,ou=People,dc=exam-
ple,dc=com" resultCode=0 etime=2.639 changeToSoftDeletedEntry=true
```

`ADD` (soft-undelete) operations

The access log displays the following.

```
[14/May/2012:09:58:16.728 -0500] ADD RESULT conn=25 op=1 msgID=1
dn="uid=user.0,ou=People,dc=example,dc=com" resultCode=0 etime=22.700
undeleteFromDN="entryUUID=ad55a34a-763f-358f-93f9-
da86f9ecd9e4+uid=user.0,
ou=People,dc=example,dc=com"
```

Audit logs

The audit log captures any `MODIFY` and `DELETE` operations of soft-deleted entries.

These changes are recorded as fully commented-out audit log entries. The audit log does not require any configuration for soft deletes.

For any soft-deleted entry, the audit log entry displays the `ds-soft-delete-entry-dn` property and its soft-deleted entry distinguished name (DN).

```
# 14/May/2012:10:57:09.054 -0500; conn=30; op=1
# ds-soft-delete-entry-dn: entryUUID=68147342-1f61-3465-8489-
3de58c532130+uid=user.2,ou=People,dc=example,dc=com
dn: uid=user.2,ou=People,dc=example,dc=com
```

```
changetype: delete
```

For any `MODIFY` changes made, the log displays the LDIF, the modifier's name, and update time.

```
# 14/May/2012:10:58:33.566 -0500; conn=33; op=1
# dn:
 entryUUID=68147342-1f61-3465-8489-3de58c532130+uid=user.2,ou=People,dc=exam-
ple,dc=com
# changetype: modify
# replace: homePhone
# homePhone: +1 003 428 0966
#-
# replace: modifiersName
# modifiersName: uid=admin,dc=example,dc=com
#-
# replace: modifyTimestamp
# modifyTimestamp: 20131010020345.546Z
```

For any undelete of a soft-deleted entry, the log displays the `ds-undelete-from-dn` attribute plus the entry unique ID, create time, and creator's name.

```
# 14/May/2012:10:59:21.754 -0500; conn=34; op=1
dn: uid=user.2,ou=People,dc=example,dc=com
changetype: add
uid: user.2
ds-undelete-from-dn:
 entryUUID=68147342-1f61-3465-8489-3de58c532130+uid=user.2,ou=Peo-
ple,dc=example,dc=com
ds-entry-unique-id:: vw1jg801S7GWrTiS3UE5DA==
createTimestamp:: 20131010181148.630Z
creatorsName: uid=admin,dc=example,dc=com
```

For hard (permanent) deletes of a soft-deleted entry, the log displays the soft-deleted entry DN that was removed.

```
# 14/May/2012:11:00:14.055 -0500; conn=36; op=1
# dn:
 entryUUID=68147342-1f61-3465-8489-3de58c532130+uid=user.2,ou=People,dc=exam-
ple,dc=com
# changetype: delete
```

Configuring the file-based audit log for soft deletes
Configure the file-based audit log for soft deletes.

Steps

1. Enable the audit log if it is disabled.

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

2. View the audit log.

   The `soft-delete-entry-audit-behavior` property is set to `commented` by default and provides additional information in comments about the soft-deleted entry that was either created or undeleted.

   ```
   # 11/May/2012:15:33:17.552 -0500; conn=13; op=1
   # ds-soft-delete-entry-dn:entryUUID=54716bfd-fbc4-3108-ac37-
   bf6b1b166e37+uid=user.15,ou=People,dc=example,dc=com
   dn: uid=user.15,ou=People,dc=example,dc=com
   ```

```
changetype: delete
```

Changelog

You can configure the changelog to capture soft-delete changes to entries so that external clients, such as PingDataSync Server, can access these changes.

The `ds-soft-delete-entry` attribute represents an entry that has been soft-deleted and is part of the source entry passed into the changelog to indicate the entry has been soft-deleted.

All soft-delete operations appear in the changelog as regular DELETE operations. When a soft delete occurs, the resulting changelog entry includes a `ds-soft-delete-entry-dn` operational attribute with the value of the soft-deleted entry DN. PingDataSync Server recognizes the `ds-soft-delete-entry-dn` attribute and does nothing with it.

The changelog backend `soft-delete-entry-included-operation` property determines whether MODIFY or DELETE operations of soft-deleted entries appear in the changelog. This property is disabled by default.

Configuring soft deletes on the changelog backend

Steps

1. To configure soft deletes on the changelog backend, run the following.

```
$ bin/dsconfig set-backend-prop \
--backend-name changelog \
--set soft-delete-entry-included-operation:delete \
--set soft-delete-entry-included-operation:modify
```

2. Run a soft-delete operation on an entry.
3. To review the changelog for the soft-deleted entry, run the following.

```
$ bin/ldapsearch --baseDN cn=changelog \
  "(objectclass=*)" "+"
```

```
dn: cn=changelog
subschemaSubentry: cn=schema
entryUUID: 9920f7e9-5a04-392a-82a8-32662d7d3863
ds-entry-checksum: 304022441
dn: changeNumber=1,cn=changelog
targetUniqueId: 94f634df-c90e-39aa-bd4a-9183c29746d0
changeTime: 20120511154141Z
ds-soft-delete-entry-dn: entryUUID=94f634df-c90e-39aa-bd4a-
9183c29746d0+uid=user.9,ou=People,dc=example,dc=com
modifyTimestamp: 20131010020345.546Z
createTimestamp:: 20131010181148.630Z
localCSN: 000001373C900852000000000003
modifiersName: uid=admin,dc=example,dc=com
entry-size-bytes: 298
subschemaSubentry: cn=schema
entryUUID: 459b06c6-89f3-307e-a515-22433eb420b6
createTimestamp: 20120511154141.431Z
modifyTimestamp: 20120511154141.431Z
ds-entry-checksum: 1157320579
```

Disabling soft deletes as a global configuration

To disable soft deletes on your Directory Server, reset the global configuration property and the remove the soft-delete policy.

About this task

From that point, PingDirectory processes all deletes as hard deletes by default. Any use of the soft-deleted options with the LDAP tools results in an error. Any existing soft-deleted entries that are present disabling after the global configuration remain in the server as latent entries.

> ⓘ **Note:**
>
> Be sure to include the LDAP bind parameters for your system when running each of these commands.

Steps

1. To reset the global configuration property, run the following **`dsconfig`** command.

```
$ bin/dsconfig set-global-configuration-prop --reset soft-delete-policy
```

2. To reset the global configuration property, run the following **`dsconfig`** command.

```
$ bin/dsconfig delete-soft-delete-policy --policy-name default-soft-delete-
policy
```

Importing and Exporting Data

The PingDirectory Server supports import or export of the database backends in LDAP Data Interchange Format (LDIF). The **`bin/import-ldif`** and **`bin/export-ldif`** tools can be used to create or export database backends for online of offline servers. The tools support options that can restrict the input or output to a subset of the entries or a subset of the attributes within entries. The tools also provide features to compress, encrypt or digitally sign the data.

Importing data

The PingDirectory Server provides initialization mechanisms to import database files. The **`import-ldif`** command-line tool imports data from an LDAP Data Interchange Format (LDIF) file. The data imported by the **`import-ldif`** command can include all or a portion of the entries (a subset of the entries or a subset of the attributes within entries or both) contained in the LDIF file. The command also supports importing data that has been compressed, encrypted or digitally signed or both.

The **`import-ldif`** utility can be run with the server offline or online. If the server is online, administrators can initiate the import from a local or remote client. The LDIF file that contains the import data must exist on the server system. During an online import, the target database repository, or backend, will be removed from service and data held in that backend will not be available to clients.

The **`import-ldif`** tool has been modified to help guard against accidental overwriting of existing backend data with the addition of the `--overwriteExistingEntries` option. This option must be present when performing an import into a backend with a branch that already contains entries (although the option is not needed if a branch contains just a single base entry).

Validating an LDIF file

About this task

Prior to importing data, you can validate an import file using the Directory Server's **`validate-ldif`** tool. When run, the tool binds to the Directory Server, locally or remotely, and validates the LDIF file to

determine whether it violates the server's schema. Those elements that do not conform to the schema will be rejected and written to standard output. You can specify the path to the output file to which the rejected entries are written and the reasons for their rejection. The **validate-ldif** tool works with regular non-compressed LDIF files or gzip-compressed LDIF files.

To process large files faster, you can also set the number of threads for validation. The tool also provides options to skip specified schema elements if you are only validating certain items, such as attributes only. Use the `--help` option to view the arguments.

To validate an LDIF file:

Steps

- Use the **validate-ldif** tool to validate an LDIF file. Make sure the server is online before running this command.

```
$ bin/validate-ldif --ldifFile /path/to/data.ldif \
  --rejectFile rejectedEntries
```

```
1 of 200 entries (0 percent) were found to be invalid.
1 undefined attributes were encountered.
Undefined attribute departmentname was encountered 1 times.
```

Computing the database cache estimate

After successful completion of an import, the **import-ldif** command lists detailed information about the database cache characteristics of the imported data set. The current server configuration is considered along with the capabilities of the underlying hardware to guide decisions for changing JVM size and database-cache-percent for the backend.

The **import-ldif** command will complete with a summary of database cache usage characteristics for the imported data set. Additional files are available in the `/logs/tools` directory that describe the database cache characteristics in more detail.

Tracking skipped and rejected entries

During import, entries can be skipped if they do not belong in the specified backend, or if they are part of an excluded base DN or filter. The `--skipFile {path}` argument can be used on the command line to indicate that any entries that are skipped should be written to a specified file. You can add a comment indicating why the entries were skipped.

Similarly, the `--rejectFile {path}` argument can be added to obtain information about which entries were rejected and why. An entry can be rejected if it violates the server's schema constraints, if its parent entry does not exist, if another entry already exists with the same DN, or if it was rejected by a plugin.

Running an offline import

You can run the **import-ldif** tool offline to import LDIF data encoded with the UTF-8 character set. This data can come from LDIF files, compressed LDIF files (GZIP format), or from data generated using a MakeLDIF template. You do not need to authenticate as an administrator when performing offline LDIF imports.

Performing an offline import

Steps

- Use the **import-ldif** command to import data from an LDIF file. Make sure the Directory Server is offline before running this command. Do not specify any connection arguments when running the command.

```
$ bin/import-ldif --backendID userRoot --ldifFile /path/to/data.ldif \
```

```
  --rejectFile /path/to/reject.ldif --skipFile /path/to/skip.ldif
```

Performing an offline LDIF import using a compressed file

Steps

- Use the **`import-ldif`** command to import data from a compressed gzip formatted file. You must also use the `--isCompressed` option to indicate that the input file is compressed. Make sure the Directory Server is offline before running this command. Do not specify any connection arguments when running the command.

```
$ bin/import-ldif --backendID userRoot --isCompressed \
  --ldifFile /path/to/data.gz --rejectFile /path/to/reject.ldif \
  --skipFile /path/to/skip.ldif
```

Performing an offline LDIF import using a MakeLDIF template

Steps

- Use the **`import-ldif`** command to import data from a MakeLDIF template, which is located in the `<server-root>/config/MakeLDIF`. Make sure the Directory Server is offline before running this command. Do not specify any connection arguments when running the command.

  The following command uses the standard data template and generates 10,000 sample entries, and then imports the file to the server.

```
$ bin/import-ldif --backendID userRoot \
  --templateFile config/MakeLDIF/example.template
```

Running an online LDIF import

Administrators can run LDIF imports while the server is online from another remote server. The online import resembles the offline import, except that you must provide information about how to connect and authenticate to the target server. You can schedule the import of an LDIF file to occur at a particular time using the `--task` and `--start YYYYMMDDhhmmss` options of the **`import-ldif`** tool.

You can also specify email addresses for users that should be notified whenever the import process completes (regardless of success or failure, or only if the import fails). To set up SMTP notifications, see *Working with the SMTP Account Status Notification Handler*.

Performing an online LDIF import

Steps

- Use the **`import-ldif`** tool to import data from an LDIF. Make sure the Directory Server is online before running this command.

```
$ bin/import-ldif --task --hostname server1 --port 389 \
  --bindDN uid=admin,dc=example,dc=com --bindPassword password \
  --backendID userRoot --ldifFile userRoot.ldif
```

Scheduling an online import

Steps

1. Use the **`import-ldif`** tool to import data from an LDIF file at a scheduled time. To specify a time in the UTC time zone, include a trailing “Z”. Otherwise, the time will be treated as a local time in the time zone configured on the server. Make sure the Directory Server is online before running this command.

```
$ bin/import-ldif --task \
  --hostname server1 \
```

```
  --port 389 \
  --bindDN uid=admin,dc=example,dc=com \
  --bindPassword password \
  --backendID userRoot \
  --ldifFile /path/to/data.ldif \
  --start 20111026010000 \
  --completionNotify import-complete@example.com \
  --errorNotify import-failed@example.com

Import task 2011102617321510 scheduled to start Oct 26, 2011 1:00:00 AM CDT
```

2. Confirm that you successfully scheduled your import task using the **`manage-tasks`** tool to view a summary of all tasks on the system.

```
$ bin/manage-tasks --summary

ID             Type   Status
-----------------------------------------------
2011102617321510 Import Waiting on start time
```

3. Use the **`manage-tasks`** tool to monitor the progress of this task. Use the task ID of the import task. If you cannot find the task ID, use the `--summary` option to view a list of all tasks scheduled on the Directory Server.

```
$ bin/manage-tasks --info 2011102617321510

Task                      Details
-------------------------------------------------------
ID                        2011102617321510
Type                      Import
Status                    Waiting on start time
Scheduled Start Time      Oct 26, 2011 1:00:00 AM CDT
Actual Start Time
Completion Time
Dependencies Failed       None
Dependency Action         None
Email Upon Completion     admin@example.com
Email Upon Error          admin@example.com

Import        Options
----------------------------
LDIF File  /path/to/data.ldif
Backend ID userRoot
```

Canceling a scheduled import

Steps

- Use the **`manage-tasks`** tool to cancel the scheduled task.

```
$ bin/manage-tasks --cancel 2011102417321510
```

Adding entries to an existing Directory Server

About this task

To add entries to an existing Directory Server while preserving operational attributes, such as `createTimestamp` or `modifiersName`, the Ignore No User Modification control must be attached to the request. The Ignore No User Modification control allows modification of certain attributes that have the No User Modification constraint. Special care should be used with this control.

The Ignore No User Modification is only applied to ADD requests. Using the control to modify an existing entry, resulting in an operational attribute change, will fail.

To append entries to an existing Directory Server:

Steps

- Use **`ldapmodify`** with the Ignore No User Modification control (i.e., the OID is 1.3.6.1.4.1.30221.2.5.5).

```
$ bin/ldapmodify --control 1.3.6.1.4.1.30221.2.5.5 \
  --filename change-record.ldif
```

Filtering data import

The **`import-ldif`** command provides a way to either include or exclude specific attributes or entries during an import. The arguments are summarized as follows:

Inclusion and Exclusion Arguments for import-ldif

Argument	Description
--includeBranch	Base DN of a branch to include in the LDIF import (can be specified multiple times)
--excludeBranch	Base DN of a branch to exclude from the LDIF import (can be specified multiple times)
--includeAttribute	Attribute to include in the LDIF import (can be specified multiple times)
--excludeAttribute	Attribute to exclude from the LDIF import (can be specified multiple times)
--includeFilter	Search filter to identify entries to include in the LDIF import (can be specified multiple times)
--excludeFilter	Search filter to identify entries to exclude from the LDIF import (can be specified multiple times)
--excludeOperational	Exclude operational attributes from the LDIF import.
--excludeReplication	Exclude replication attributes from the LDIF import.
--excludeSoftDelete	Exclude soft delete entries from the LDIF import.

Exporting data

The PingDirectory Server **`export-ldif`** command-line tool exports data from Directory Server backend to an LDAP Data Interchange Format (LDIF) file. The tool must be run in the non-task based mode, which implies that it works outside of the server JVM process. The **`export-ldif`** must be run without connection or task arguments while the server is either online or offline. This tool exports a point-in-time snapshot of the backend which is guaranteed to provide a consistent state of the database, in LDIF, which can be reimported with **`import-ldif`** if necessary.

The data exported by **`export-ldif`** can include all or a portion of the entries (a subset of the entries or a subset of the attributes within entries or both) contained in the backend. This is accomplished by specifying branches, filters, and attributes to include or exclude. The exported LDIF can be compressed, encrypted or digitally signed.

ⓘ **Note:** LDIF exports can be configured as recurring tasks with **`dsconfig create-recurring-task`**, and then scheduled to run when added to a recurring task chain.

Performing an export

Steps

- Use the **`export-ldif`** command to export data to an LDIF file.

```
$ bin/export-ldif --backendID userRoot --ldifFile userRoot.ldif
```

Performing an export from specific branches

Steps

- Use the **`export-ldif`** command to export data to an LDIF file under a specific branch from the userRoot backend of the local Directory Server into a compressed file. The command also excludes operational attributes from the exported data and wraps long lines at column 80.

```
$ bin/export-ldif --backendID userRoot --ldifFile userRoot.ldif.gz --
compress \
  --includeBranch ou=people,dc=example,dc=com --excludeOperational \
  --wrapColumn 80
```

Encrypting LDIF exports and signing LDIF files

The Directory Server provides features to encrypt data during an LDIF export using the **`export-ldif`** `--encryptLDIF` option and to allow the encrypted LDIF file to be imported onto the same instance or another server in the same replication topology using the **`import-ldif`** tool. A **`--doNotEncrypt`** argument can be used to force an LDIF export to be unencrypted, even if automatic encryption is enabled. The **`--maxMegabytesPerSecond`** argument can be used to impose a limit on the rate at which the LDIF file may be written to disk.

The **`export-ldif`** tool can be used with the **`--promptForEncryptionPassphrase`**, **`--encryptionPassphraseFile`**, and **`--encryptionSettingsDefinitionID`** arguments to specify which key to use for encrypting the export. The **`import-ldif tool`** will automatically detect encryption and compression, and have **`--promptForEncryptionPassphrase`**, **`--encryptionPassphraseFile`** options as well.

The Directory Server also provides an additional argument that digitally signs the contents of the LDIF file, which ensures that the content has not been altered since the export. To digitally sign the contents of the exported LDIF file, use the **`export-ldif`** `--sign` option. To allow a signed LDIF file to be imported onto the same instance or another server in the same topology, use the **`import-ldif`** `--isSigned` option.

Note that there is not much added benefit to both signing and encrypting the same data, since encrypted data cannot be altered without destroying the ability to decrypt it.

Encrypting an LDIF export

Steps

- Run **`export-ldif`** tool with the **`--encryptLDIF`** option to encrypt the data during an export to an output LDIF file. The following command runs an offline export of the userRoot backend, and encrypts the file when written to an output file called `data.ldif`.

```
$ bin/export-ldif --backendID userRoot --ldifFile /path/to/data.ldif \
  --encryptLDIF
```

Importing an encrypted LDIF file

About this task

An encrypted LDIF file can be imported into the same instance from which it was exported, or into any other server in the same replication topology with that instance. You cannot import an encrypted LDIF file into a server that is not in some way connected to the instance from which it was exported.

Steps

- Run the `import-ldif` tool to import the encrypted LDIF file from the previous example. The command imports the `data.ldif` file, decrypts the contents while overwriting the existing contents to the `userRoot` backend. The tool automatically determines encryption and compression, and it can automatically identify the correct key for exports that were encrypted with a key obtained from an encryption settings definition or an internal topology key.

```
$ bin/import-ldif --backendID userRoot --ldifFile /path/to/data.ldif \
  --overwriteExistingEntries
```

Signing an export

Steps

- Run **`export-ldif`** tool with the **`--sign`** option to digitally sign the data during an export to an output LDIF file. The following command runs an offline export of the `userRoot` backend, and signs the content when written to an output file called `data.ldif`.

```
$ bin/export-ldif --backendID userRoot \
  --ldifFile /path/to/data.ldif --sign
```

Importing a signed LDIF file

Steps

- Run the **`import-ldif`** tool to import the signed LDIF file from the previous example. The command imports the `data.ldif` file, checks the signature of the contents while overwriting the existing contents to the `userRoot` backend. The command requires the `--isSigned` option, which instructs the tool that the contents of the LDIF file is signed.

```
$ bin/import-ldif --backendID userRoot \
  --ldifFile /path/to/data.ldif \
  --overwriteExistingEntries --isSigned
```

Filtering data exports

The **`export-ldif`** command analogous arguments to the **`import-ldif`** tool to provide a way to either include or exclude specific attributes or entries during an export. The arguments are summarized as follows:

Inclusion and Exclusion Arguments for export-ldif

Argument	Description
--includeBranch	Base DN of a branch to include in the LDIF export (can be specified multiple times)
--excludeBranch	Base DN of a branch to exclude from the LDIF export (can be specified multiple times)
--includeAttribute	Attribute to include in the LDIF export (can be specified multiple times)

Argument	Description
--excludeAttribute	Attribute to exclude from the LDIF export (can be specified multiple times)
--includeFilter	Filter to identify entries to include in the LDIF export (can be specified multiple times)
--excludeFilter	Filter to identify entries to exclude from the LDIF export (can be specified multiple times)
--excludeOperational	Exclude operational attributes from the LDIF export.
--excludeReplication	Exclude replication attributes from the LDIF export.
--excludeSoftDelete	Exclude soft delete entries from the LDIF export.

Scrambling data files

The Directory Server **transform-ldif** tool provides backward compatibility with the former **scramble-ldif** tool, with additional functionality for configuring input and output files. The **transform-ldif** tool reads data from one or more source LDIF files and writes the transformed data to a single output file.

Using this tool to scramble data, enables obscuring the values of certain attributes so that it is difficult to determine the original values in the source data, while also preserving the characteristics of the associated attribute syntax. This process is repeatable, so that if the same value appears multiple times, it will yield the same scrambled representation each time. Scrambling can be applied to both LDIF entries and LDIF change records.

The process of scrambling data is not the same as encryption. It should only be used to provide simple obfuscation of data. The following are general guidelines for scrambling attributes:

- If the attribute is **userPassword** and its value starts with a scheme name surrounded by curly braces, such as "{SSHA256}XrgyNdl3fid7KYdhd/Ju47KJQ5PYZqlUlyzxQ28f/QXUnNd9fupj9g==", the scheme name will be left unchanged and the rest of the value will be treated like a generic string.
- If the attribute is **authPassword** and its value contains at least two dollar signs, such as "SHA256$QGbHtDCi1i4=$8/X7XRGaFCovC5mn7ATPDYlkVoocDD06Zy3lbD4AoO4=", the portion up to the first dollar sign (which represents the name of the encoding scheme) is preserved and the remainder of the value is treated like a generic string.
- If an attribute has a Boolean syntax, the scrambled value will be either **TRUE** or **FALSE**. The determination to use a value of **TRUE** or **FALSE** is random, so scrambling Boolean values is not repeatable. By randomizing the scrambling for Boolean values, the syntax and obfuscation of the original value is preserved.
- If an attribute has a distinguished name syntax (or a related syntax, such as a name and optional UID), scrambling is applied to the values of RDN components for any attributes to be scrambled. For example, if the tool is configured to scramble both the **member** and **uid** attributes, and an entry has a member attribute with a value of "uid=john.doe,ou=People,dc=example,dc=com", that member value will be scrambled in a way that only obscures the "john.doe" portion but leaves the attribute names and all values of non-scrambled attributes intact.
- If an attribute has a generalized time syntax, that value is replaced with a randomized timestamp using the same format (the same number of digits and the same time zone indicator). The randomization will be over a time range that is double the difference between the time the **transform-ldif** tool was launched and the timestamp to be scrambled. For values where that time difference is less than one day, one day will be added to the difference before it is doubled.
- If an attribute has an integer, numeric string, or telephone number syntax, scrambling is only applied to numeric digits while all other characters are left intact. If there are multiple digits, then the first digit will be nonzero.

- If an attribute has an octet string syntax, it is scrambled as follows:
  - Each byte that represents a lowercase ASCII letter is replaced with a randomly-selected lowercase ASCII letter.
  - Each byte that represents a uppercase ASCII letter is replaced with a randomly-selected uppercase ASCII letter.
  - Each byte that represents an ASCII digit is replaced with a randomly-selected ASCII digit.
  - Each byte that represents a printable ASCII symbol is replaced with a randomly-selected printable ASCII symbol.
  - Each byte that represents an ASCII control character is replaced with a randomly-selected ASCII letter, digit, or symbol.
  - Each non-ASCII byte will be replaced with a randomly-selected non-ASCII byte.
- If an attribute has a value that represents a valid JSON object, the resulting value will also be a JSON object. All field names will be left intact, and only the values of those fields may be scrambled. If the **`--scrambleJSONField`** argument is provided, only the specified fields will have values scrambled. Otherwise, the values of all fields will be scrambled. Field values are scrambled as follows:
  - Null values are not scrambled.
  - Boolean values are replaced with randomly-selected Boolean values. As with attributes with a Boolean syntax, these values are non-repeatable.
  - Number values will have only their digits replaced with randomly-selected digits and all other characters (minus sign, decimal point, exponentiation indicator) are left unchanged.
  - String values will be replaced with a randomly-selected generic string.
  - Array values have scrambling applied as appropriate for each value in the array. If the array field itself should be scrambled, then all values in the array are scrambled. Otherwise, only JSON objects contained inside the array have scrambling applied to appropriate fields.
  - JSON values have scrambling applied as appropriate for their fields.
- If an attribute does not match any of the previous criteria, it is scrambled as follows:
  - Each lowercase ASCII letter is replaced with a randomly-selected lowercase ASCII letter.
  - Each uppercase ASCII letter replaced with a randomly-selected uppercase ASCII letter.
  - Each ASCII digit is replaced with a randomly-selected ASCII digit.
  - All other characters are left unchanged.

The following example reads from an LDIF file named **`original.ldif`**, scrambles the values of the **`telephoneNumber`**, **`mobile`**, and **`homeTelephoneNumber`** attributes, and writes the results to **`scrambled.ldif`**:

```
$ bin/transform-ldif --sourceLDIF original.ldif \
  --targetLDIF scrambled.ldif \
  --scrambleAttribute telephoneNumber \
  --scrambleAttribute mobile \
  --scrambleAttribute homeTelephoneNumber \
  --randomSeed 0
```

Backing Up and Restoring Data

The PingDirectory Server provides efficient **`backup`** and **`restore`** command-line tools that support full and incremental backups. The backups can also be scheduled using the UNIX-based cron scheduler or using the Directory Server's Task-based scheduler.

About backing up and restoring data

Administrators should have a comprehensive backup strategy and schedule that comprise of daily, weekly, and monthly backups including incremental and full backups of the directory server data, configuration, and

backends. Administrators should also have a backup plan for the underlying file system. This dual purpose approach provides excellent coverage in the event that a server database must be restored for any reason.

ⓘ **Note:** Backups can be configured as recurring tasks with **dsconfig create-recurring-task**, and then scheduled to run when added to a recurring task chain.

The PingDirectory Server provides efficient **backup** and **restore** command-line tools that support full and incremental backups. The backups can also be scheduled using the UNIX-based cron scheduler or using the Directory Server's Task-based scheduler. The Directory Server can run backups with the server online while processing other requests, so that there is no need to shut down the server or place it in read-only mode prior to starting a backup.

If you back up more than one backend, the **backup** tool creates a subdirectory below a specified backup directory for each backend. If you back up only a single backend, then the backup files will be placed in the specified directory. A single directory can only contain files from one backend, so that you cannot have backup files from multiple different backends in the same backup directory.

When performing a backup, the server records information about the current state of the server and backend, including the server product name, the server version, the backend ID, the set of base DNs for the backend, and the Java class used to implement the backend logic. For JE backends, the backup descriptor also includes information about the Berkeley DB JE version and information about the attribute and VLV indexes that have been defined.

When restoring a backup, the server compares the descriptor obtained from the backup with the current state of the server and backend. If any problems are identified, the server generates warnings or errors. The administrator can choose to ignore the warnings with the `ignoreCompatibilityWarnings` option to the **restore** tool, whereas errors will always cause the restore to fail. For example, when restoring a *newer* backup into an older version of the server, a warning will be generated. When restoring an *older* backup into a new version of the server, no warning will be generated, but because the `config` and `schema` backends require special handling, the server generates an error if the server versions do not match exactly (major, minor, point, and patch version numbers).

Both the **backup** and **restore** tools provide encryption options **--promptForEncryptionPassphrase**, **--encryptionPassphraseFile**, and **--encryptionSettingsDefinitionID** that can be used to specify which key to use for encrypting the backup. For backups encrypted with an encryption settings definition or an internal topology key, the server will automatically determine the correct key. Or, the **--doNotEncrypt** argument can be used to force a backup to be unencrypted even if automatic encryption is enabled.

If needed, the **--maxMegabytesPerSecond** argument can be used to impose a limit on the rate at which the backup may be written to disk.

Retaining backups

The backup tool can be used with either the **--retainPreviousFullBackupCount** or **--retainPreviousFullBackupAge** arguments to identify which previous backups should be preserved. Any other backups in that directory will be removed. A new backup will always be preserved. If the new backup is an incremental backup, then any other backups it depends on will also be preserved. However, older backups in the same directory are eligible to be removed.

If the **--retainPreviousFullBackupCount** argument is provided, that number of the most recent previous full backups will be preserved (along with any incremental backups that depend on them). Any other previous full backups (and their dependent incremental backups) can be removed. If the **--retainPreviousFullBackupAge** argument is provided, its value must be a duration represented as an integer followed by a time unit. Any full backups (and their dependent incremental backups) created longer ago than that duration will be eligible to be removed. If both the **--retainPreviousFullBackupCount** and **--retainPreviousFullBackupAge** arguments are provided, then only backups that don't satisfy either condition will be deleted. A value of zero can be

specified for the **--retainPreviousFullBackupCount** argument so that only the most recent backup is preserved (along with its dependencies), and all previous backups will be removed.

ⓘ **Note:** The **remove-backup** tool also supports the **--retainFullBackupCount** and **--retainFullBackupAge** arguments to delete any backups outside the provided retention criteria.

Listing the available backups on the system

Steps

- Use the **restore** tool to list the backups in a `backup` directory.

```
$ bin/restore --listBackups --backupDirectory /mybackups
```

```
[13:26:21] The console logging output is also available in '/
ds/PingDirectory/logs/ tools/restore.log'

Backup ID:         20120212191715Z
Backup Date:       12/Feb/2012:13:17:19 -0600
Is Incremental:    false
Is Compressed:     false
Is Encrypted:      false
Has Unsigned Hash: false
Has Signed Hash:   false
Dependent Upon:    none
Backup ID:         20120212192411Z
Backup Date:       12/Feb/2012:13:24:16 -0600
Is Incremental:    true
Is Compressed:     false
Is Encrypted:      false
Has Unsigned Hash: false
Has Signed Hash:   false
Dependent Upon:    20120212191715Z
```

Backing up all backends

Steps

- Use **backup** to save the all of the server's backends. The optional `--compress` option can reduce the amount of space that the backup consumes, but can also significantly increase the time required to perform the backup.

```
$ bin/backup --backUpAll --compress --backupDirectory /path/to/backup
```

Backing up a single backend

Steps

- Go to the server root directory, and use the **backup** tool to save the single backend, userRoot.

```
$ bin/backup --backendID userRoot --compress --backupDirectory /path/to/
backup
```

Performing an offline restore

Steps

- Use the **`restore`** command to restore the userRoot backend. Only a single backend can be restored at a time. The Directory Server must be shut down before performing an offline restore.

```
$ bin/restore --backupDirectory /path/to/backup/userRoot
```

> ⓘ **Note:** The server root directory should never be restored from a file system backup or snapshot.

Assigning an ID to a backup

Steps

- Go to the server root directory, and use the backup tool to save the single backend, userRoot. The following command assigns the backup ID "weekly" to the userRoot backup. The backup file appears under `backups/userRoot` directory as `userRoot-backup-weekly`.

```
$ bin/backup --backupDirectory /path/to/backups/userRoot \
  --backendID userRoot --backupID weekly
```

Running an incremental backup on all backends

About this task

The Directory Server provides support for incremental backups, which backs up only those items that have changed since the last backup (whether full or incremental) on the system, or since a specified earlier backup. Incremental backups must be placed in the same backup directory as the full backup on which they are based.

Not all backends support incremental backups. If a backend does not support incremental backups, use of the **`--incremental`** option will have no effect, and a full backup will be taken.

Steps

- The following command runs an incremental backup on all backends based on the most recent backup:

```
$ bin/backup --backUpAll --incremental --backupDirectory /path/to/backup
```

Running an incremental backup on a single backend

Steps

- Go to the server root directory, and use **`backup`** to save the single backend, userRoot.

```
$ bin/backup --backendID userRoot --incremental --backupDirectory /path/to/
backup
```

Running an incremental backup based on a specific prior backup

Steps

- You can run an incremental backup based on a specific prior backup that is not the most current version on the system. To get the backup ID, use the **`restore`** `--listBackups` command (see below).

```
$ bin/backup --backUpAll --incremental --backupDirectory /path/to/backup \
  --incrementalBaseID backup-ID
```

Restoring an incremental backup

About this task

The process for restoring an incremental backup is exactly the same as the process for restoring a full backup for both the online and offline restore types. The **`restore tool`** will automatically ensure that the full backup and any intermediate incremental backups are restored first before restoring the final incremental backup. The tool will not restore any files in older backups that are no longer present in the final data set.

Scheduling an online backup

About this task

You can schedule a backup to run as a Task by specifying the timestamp with the `--task` and `--start` options. The option is expressed in "YYYYMMDDhhmmss'" format. If the option has a value of "0" then the task is scheduled for immediate execution. You cannot run recurring tasks, so daily operations must be run using cron or through some system that can submit the task.

For online (remote) backups, the backup operation can be conducted while PingDirectory Server is online if you provide information about how to connect and to authenticate to the target Directory Server.

Steps

- You can schedule the backup to occur at a specific time using the Task-based `--start YYYYMMDDhhmmss` option. To specify a time in the UTC time zone format, add a trailing “Z” to the time. Otherwise, the time will be treated as a local time in the time zone configured on the server.

```
$ bin/backup --backUpAll --task --start 20111025010000 \
  --backupDirectory /path/to/backup --completionNotify admin@example.com \
  --errorNotify admin@example.com

Backup task 2011102500084110 scheduled to start Oct 28, 2011 1:00:00 AM CDT
```

To Schedule an Online Restore

About this task

By providing connection and authentication information (and an optional start time), the restore can be performed via the Tasks subsystem while the server is online. The Tasks subsystem allows you to schedule certain operations, such as **`import-ldif`**, **`backup`**, **`restore`**, **`start-server`**, and **`stop-server`**. You can schedule a restore to run as a Task by specifying the timestamp with the `--task` and `--start` options. The option is expressed in "YYYYMMDDhhmmss'" format. If the option has a value of "0" then the task is scheduled for immediate execution. You cannot run recurring tasks, so daily operations must be run using cron or through some system that can submit the task.

Steps

- The backend that is being restored will be unavailable while the restore is in progress. To specify a time in the UTC time zone, add a trailing “Z” to the time. Otherwise, the time will be treated as a local time in the configured time zone on the server.

```
$ bin/restore --task --start 20111025010000 \
  --backupDirectory /path/to/backup/userRoot \
  --completionNotify admin@example.com --errorNotify admin@example.com
```

Encrypting a backup

Steps

- Go to the server root directory, and use the **backup** tool to backup up the single backend, userRoot and encrypt it with the **--encrypt** option.

```
$ bin/backup --encrypt --backendID userRoot --compress --backupDirectory /
path/to/backup
```

Signing a hash of the backup

Steps

- Go to the server root directory, and use the **backup** tool to backup up the single backend, userRoot. Use the **-signHash** option to generate a hash of the backup contents and digitally sign the hash of the backup contents. If you want to generate only a hash of the backup contents, run backup with the **--hash** option.

```
$ bin/backup --signHash --backupDirectory backups/userRoot --backendID
 userRoot \
  --backupDirectory /path/to/backup
```

Restoring a backup

Steps

- Go to the server root directory, and use the **restore** tool to restore a backup. The **backup** tool uses a descriptor file to access property information used for the backup, indicating if the backup was compressed, signed and/or encrypted.

```
$ bin/restore --backupDirectory /path/to/backup
```

Moving or restoring a user database

Part of any disaster recovery involves the restoration of the user database from one server to another. You should have a well-defined backup plan that takes into account whether or not your data is replicated among a set of servers. The plan is the best insurance against significant downtime or data loss in the event of unrecoverable database issue.

ⓘ **Note:** The server root directory should never be restored from a file system backup or snapshot. External backup methods (for example, VM snapshots) are not recommended, especially if data was corrupted during a transaction.

Keep in mind the following general points about database recovery:

- **Regular Backup from Local Replicated Directory Server** . Take a backup from a local replicated directory server and restore to the failed server. This will be more recent than any other backup you have.
- **Restore the Most Recent Backup**. Restore the most recent backup from a local server. In some cases, this may be preferred over taking a new backup if that would adversely impact performance of the server being backed up although it will take longer for replication to play back changes.
- **Contact Support**. If all else fails, contact your authorized support provider and they can work with you (and possibly in cooperation with the Oracle Berkeley DB JE engineers) to try a low-level recovery, including tools that attempt to salvage whatever data they can obtain from the database.

Comparing the data in two Directory Servers

The PingDirectory Server provides an **`ldap-diff`** tool to compare the data on two LDAP servers to determine any differences that they may contain. The differences are identified by first issuing a subtree search on both servers under the base DN using the default search filter `(objectclass=*)` to retrieve the DNs of all entries in each server. When the tool finds an entry that is on both servers, it retrieves the entry from each server and compares all of its attributes. The tool writes any differences it finds to an LDIF file in a format that could be used to modify the content of the source server, so that it matches the content of the target server. Any non-synchronized entries can be compared again for a configurable number of times with an optional pause between each attempt to account for replication delays.

You can control the specific entries to be compared with the `--searchFilter` option. In addition, only a subset of attributes can be compared by listing those attributes as trailing arguments of the command. You can also exclude specific attributes by prepending a ^ character to the attribute. (On Windows operating systems, excluded attributes must be quoted, for example, "`^attrToExclude`".) The `@objectClassName` notation can be used to compare only attributes that are defined for a given objectclass.

The **`ldap-diff`** tool can be used on servers actively being modified by checking differing entries multiple times without reporting false positives due to replication delays. By default, it will re-check each entry twice, pausing two seconds between checks. These settings can be configured with the `--numPasses` and `--secondsBetweenPass` options. If the utility cannot make a clean comparison on an entry, it will list any exceptions in comments in the output file.

The Directory Server user specified for performing the searches must be privileged enough to see all of the entries being compared and to issue a long-running, unindexed search. For the Directory Server, the out-of-the-box `cn=Directory Manager` user has these privileges, but you can assign the necessary privileges by setting the following attributes in the user entry:

```
ds-cfg-default-root-privilege-name: unindexed-search
ds-cfg-default-root-privilege-name: bypass-acl
ds-rlim-size-limit: 0
ds-rlim-time-limit: 0
ds-rlim-idle-time-limit: 0
ds-rlim-lookthrough-limit: 0
```

The `ldap-diff` tool tries to make efficient use of memory, but it must store the DNs of all entries in memory. For Directory Servers that contain hundreds of millions of entries, the tool might require a few gigabytes of memory. If the progress of the tool slows dramatically, it might be running low on memory. The memory used by the `ldap-diff` tool can be customized by editing the `ldap-diff.java-args` setting in the `config/java.properties` file and running the **`dsjavaproperties`** command.

If you do not want to use a subtree search filter, you can use an input file of DNs for the source, target, or both. The format of the file can accept various syntaxes for each DN:

```
dn: cn=this is the first dn
dn: cn=this is the second dn and it is wrapped cn=this is the third dn
# The following DN is base-64 encoded dn::
Y249ZG9uJ3QgeW91IGhhdmUgYmV0dGVyIHRoaW5ncyB0byBkbyB0aGFuIHNlZSB3aGF0IHRoaXMgc2F5cw==
# There was a blank line above dn: cn=this is the final entry.
```

ⓘ **CAUTION:** Do not manually update the servers when the tool identifies differences between two servers involved in replication. First contact your authorized support provider for explicit confirmation, because manual updates to the servers risk introducing additional replication conflicts.

Comparing two Directory Servers using ldap-diff

Steps

1. Use **`ldap-diff`** to compare the entries in two Directory Server instances. Ignore the `userpassword` attribute due to the one-way password hash used for the password storage scheme.

```
$ bin/ldap-diff --outputLDIF difference.ldif \
  --sourceHost server1.example.com --sourcePort 1389 \
  --sourceBindDN "cn=Directory Manager" --sourceBindPassword secret1 \
  --targetHost server2.example.com --targetPort 2389 \
  --targetBindDN "cn=Directory Manager" --targetBindPassword secret2 \
  --baseDN dc=example,dc=com --searchFilter "(objectclass=*)"
 "^userpassword"
```

2. Open the output file in a text editor to view any differences. The file is set up so that you can re-apply the changes without any modification to the file contents. The file shows any deletes, modifies, and then adds from the perspective of the source server as the authoritative source.

```
# This file contains the differences between two LDAP servers.
#
# The format of this file is the LDIF changes needed to bring server
# ldap://server1.example.com:1389 in sync with server
# ldap://server2.example.com:2389.
#
# These differences were computed by first issuing an LDAP search at both
# servers under base DN dc=example,dc=com using search filter
 (objectclass=*)
# and search scope SUB to first retrieve the DNs of all entries. And then
 each
# entry was retrieved from each server and attributes: [^userpassword] were
# compared. # # Any entries that were out-of-sync were compared a total of 3
 times
# waiting a minimum of 2 seconds between each attempt to account for
 replication
# delays.
#
# Comparison started at [24/Feb/2010:10:34:20 -0600]
# The following entries were present only on ldap://server2.example.com:2389
 and
# need to be deleted. This entry existed only on ldap://
server1.example.com:1389
# Note: this entry might be incomplete. It only includes attributes:
# [^userpassword]dn: uid=user.200,ou=People,dc=example,dc=com
# objectClass: person
# objectClass: inetOrgPerson
... (more attributes not shown) ...
# st: DC
dn: uid=user.200,ou=people,dc=example,dc=com
changetype: delete

# The following entries were present on both servers but were out of sync.

dn: uid=user.199,ou=people,dc=example,dc=com
changetype: modify
add: mobile
mobile: +1 300 848 9999
-
delete: mobile
mobile: +1 009 471 1808

# The following entries were missing on ldap://server2.example.com:2389 and
 need
```

```
# to be added. This entry existed only on ldap://server2.example.com:2389
# Note: this entry might be incomplete. It only includes attributes:

# [^userpassword]
dn: uid=user.13,ou=People,dc=example,dc=com
changetype: add
objectClass: person
objectClass: inetOrgPerson
... (more attributes not shown) ...
# Comparison completed at [24/Feb/2010:10:34:25 -0600]
```

Comparing configuration entries using config-diff

Steps

- Use **`config-diff`** to compare Directory Server configurations and produce a **`dsconfig`** batch file needed to bring the source inline with the target.

  The following example compares the current configurations of server1 and server2. The changes necessary to bring server1's configuration inline with server2 are written to the console. The same credentials are used to connect to both servers.

  ```
  $ bin/config-diff --sourceHost server1 --sourceBindDN "cn=Directory Manager"
   \
    --sourceBindPassword password --targetHost server2
  ```

  For more information about runtime options, see the *Command Line Tool Reference* or the **`config-diff`** tool help.

Comparing entries using source and target DN files

Steps

- Use **`ldap-diff`** to compare the entries in two Directory Server instances. In the following example, the utility uses a single DN input file for the source and target servers, so that no search filter is used. Ignore the `userpassword` attribute due to the password storage scheme that uses a one-way hashing algorithm.

  ```
  $ bin/ldap-diff --outputLDIF difference.ldif \
    --sourceHost server1.example.com --sourcePort 1389 \
    --sourceBindDN "cn=Directory Manager" --sourceBindPassword secret1 \
    --targetHost server2.example.com --targetPort 2389 \
    --targetBindDN "cn=Directory Manager" --targetBindPassword secret2 \
    --baseDN "dc=example,dc=com" --sourceDNsFile input-file.ldif \
    --targetDNsFile input-file.ldif "^userpassword"
  ```

Comparing Directory Servers for missing entries only using ldap-diff

Steps

- Use **`ldap-diff`** to compare two Directory Servers and return only those entries that are missing on one of the servers using the `--missingOnly` option, which can significantly reduce the runtime for this utility.

  ```
  $ bin/ldap-diff --outputLDIF difference.ldif \
    --sourceHost server1.example.com --sourcePort 1389 \
    --sourceBindDN "cn=Directory Manager" --sourceBindPassword secret1 \
    --targetHost server2.example.com --targetPort 2389 \
    --targetBindDN "cn=Directory Manager" --targetBindPassword secret2 \
    --baseDN dc=example,dc=com --searchFilter "(objectclass=*)"
   "^userpassword" \
  ```

```
  --missingOnly
```

Reverting or replaying changes

The PingDirectory Server provides support for an audit logger that records information about the changes to data within the server. The data is formatted as LDIF, and it can be replayed with tools such as **ldapmodify** or **parallel-update**. The data also includes information encoded as comments that provide additional context about the changes. By default, the log records the changes as requested by clients, but it can also log the changes in reversible form so that they can be undone.

This audit logger can be useful for the following scenarios:

- If one or more undesirable changes have been made (for example, by a malicious or defective client), it can be used to obtain the necessary changes to revert those operations.
- If a catastrophic loss of all servers in the topology occurs that leaves an audit log available with newer data than any backup or LDIF export (for example, concurrent database corruption across all instances), it can be used to recover changes that may not otherwise be available.
- It can be useful for automating the process of identifying changes made in one topology that can be replayed into another topology (for example, to replay production changes into an isolated server or topology for testing purposes or to attempt to reproduce a problem).
- It can be useful for analytics and reporting purposes.

To assist with these and other uses, the LDAP SDK for Java provides an API for consuming, parsing, and reverting audit log messages. This API can be used for the analytics and reporting. Also available is the **extract-data-recovery-log-changes** tool that can extract audit log changes matching a specified set of criteria so that they can be replayed, either as they were originally processed or in a reversible form that makes it possible to revert those changes.

The Data Recovery Log

The **setup** tool automatically creates an audit logger for data recovery purposes in **logs/data-recovery**. The log is always compressed, and it will be encrypted if data encryption is enabled within the server. The logger has the following properties:

- Log files are written into the **logs/data-recovery** directory so that they are isolated from other log files. The active log file is named **data-recovery.gz.encrypted**, while rotated files are named **data-recovery.{timestamp}.gz.encrypted**.
- The log files are gzip-compressed. If data encryption is enabled, they are encrypted with a key obtained from the server's preferred encryption settings definition.
- Each log file contains no more than 10 MB of data, and is rotated after 24 hours. Keeping the log files small ensures that the entire contents of a log file will easily fit into the **extract-data-recovery-log-changes** tool's memory.
- The server will retain rotated data recovery log files for no more than one week. However, as a safeguard against consuming too much disk space in periods of extremely heavy and prolonged write activity, the server will also retain no more than 1,000 data recovery log files for a maximum of 500 MB of disk space.
- Changes are logged in reversible form and include the authentication and authorization identity of the requester, as well as the IP address. If present, the log message includes details from any intermediate client request control included in the request, which may provide information about the downstream client.

The extract-data-recovery-log-changes Tool

The **extract-data-recovery-log-changes** tool creates an LDIF file (compressed and encrypted by default) with a specified subset of changes from the server's data recovery log. That LDIF file can then be applied to the server using either the **ldapmodify** or parallel-update. Before applying the changes, the output file can be decrypted and examined to ensure that the changes it contains look correct. This tool can be useful for disaster recovery.

The **`extract-data-recovery-log-changes`** tool provides arguments for input and output of the extracted changes, including encryption settings, location, and compression.

The direction of whether changes should be extracted in forward mode or reverse mode is also configured. In forward mode (replay), the audit log messages are traversed from oldest to newest, and extracted changes are presented as they were originally requested. In reverse mode (revert), the audit log messages are traversed from newest to oldest, and extracted changes will be converted to a form that will revert the original changes. Regardless of the direction chosen, additional arguments enable identifying the changes to extract by time, requester address or DN, connection ID, origin, content type, or alterations. The following is a sample command to revert all changes by user **`uid=malicious,ou=People,dc=example,dc=com`** between noon and 2 pm on October 15, 2018:

```
$ bin/extract-data-recovery-log-changes \
  --auditLogFile logs/data-recovery/data-
recovery.201810161234.567.gz.encrypted \
  --outputFile revert-malicious-user-changes.ldif \
  --direction revert \
  --startTime 201810151200.000 \
  --endTime 201810151359.999 \
  --includeAuthorizationDN "uid=malicious,ou=People,dc=example,dc=com"
```

Working with Groups

LDAP groups are special types of entries that represent collections of users. Groups are often used by external clients, for example, to control who has access to a particular application or features. They may also be used internally by the server to control its behavior. For example, groups can be used by the access control, criteria, or virtual attribute subsystems.

The specific ways in which clients create and interact with a particular group depends on the type of group being used. In general, there are three primary ways in which clients attempt to use groups:

- To determine whether a specified user is a member of a particular group.
- To determine the set of groups in which a specified user is a member.
- To determine the set of all users that are members of a particular group.

This chapter provides an overview of Directory Server groups concepts and provides procedures on setting up and querying groups in the Directory Server.

Overview of groups

The Directory Server provides the following types of groups:

- **Static Groups**. A static group is an entry that contains an explicit list of member or uniquemember attributes, depending on its particular structural object class. Static groups are ideal for relatively small, infrequently changing elements. Once the membership list grows, static groups become more difficult to manage as any change in a member base DN must also be changed in the group. Static groups use one of three structural object classes: `groupOfNames`, `groupOfUniqueNames`, and `groupOfEntries`.

  The Directory Server also supports nested groups, in which a parent group entry contains child attributes whose DNs reference another group. Nested groups are a flexible means to organize entries that provide inherited group membership and privileges. To maintain good performance throughput, a group cache is enabled by default. The cache supports static group nesting that includes other static, virtual static, and dynamic groups.
- **Dynamic Groups**. A dynamic group has its membership list determined by search criteria using a LDAP URL. Dynamic groups solve the scalability issues encountered for static groups as searches are efficient, constant-time operations. However, if searches range over a very large set of data, performance could be affected.

- **Virtual Static Groups**. A virtual static group is a combination of both static and dynamic groups, in which each member in a group is a virtual attribute that is dynamically generated when invoked. Virtual static groups solve the scalability issues for clients that can only support static groups and are best used when the application targets a search operation for a specific member. Virtual static groups are not good for applications that need to retrieve the entire membership list as the process for constructing the entire membership list can be expensive.

About the isMemberOf and isDirectMemberOf virtual attribute

The existence of both static, nested, dynamic, and virtual static groups can make it unnecessarily complex to work with groups in the server, particularly because the ways you interact with them are so different. And the fact that static groups can use three different structural object classes (not counting the auxiliary class for virtual static groups) does not make things any easier.

To make group operations simpler, the PingDirectory Server provides the ability to generate either an `isMemberOf` and `isDirectMemberOf` virtual attributes in user entries. These attributes dramatically simplify the process for making group-related determinations in a manner that is consistent across all types of groups.

The value of the `isMemberOf` virtual attribute is a list of DNs of all groups (including static, nested, dynamic, and virtual static groups) in which the associated user is a member. The value of the `isDirectMemberOf` virtual attribute is a subset of the values of `isMemberOf`, which represents the groups for which the entry is an explicit or direct member. Both are enabled by default.

Because the `isMemberOf` and `isDirectMemberOf` are operational attributes, only users who specifically have been granted the privilege can see it. The default set of access control rules do not allow any level of access to user data. The only access that is granted is what is included in user-defined access control rules, which is generally given to a `uid=admin` administrator account. It is always a best practice to restrict access to operational and non-operational attributes to the minimal set of users that need to see them. The root bind DN, `cn=Directory Manager`, has the privilege to view operational attributes by default.

To determine whether a user is a member of a specified group using the `isMemberOf` virtual attribute, simply perform a base-level search against the user's entry with an equality filter targeting the `isMemberOf` attribute with a value that is the DN of the target group. The following table illustrates this simple base-level search:

Base DN	uid=john.doe,ou=People,dc=example,dc=com
Scope	base
Filter	(isMemberOf=cn=Test Group,ou=Groups,dc=example,dc=com)
Requested Attributes	1.1

If this search returns an entry, then the user is a member of the specified group. If no entry is returned, then the user is not a member of the given group.

To determine the set of all groups in which a user is a member, simply retrieve the user's entry with a base-level search and include the `isMemberOf` attribute:

Base DN	uid=john.doe,ou=People,dc=example,dc=com
Scope	base
Filter	(objectclass=*)
Requested Attributes	isMemberOf

To determine the set of all members for a specified group, issue a subtree search with an equality filter targeting the `isMemberOf` attribute with a value that is the DN of the target group and requesting the attributes you wish to have for member entries:

Base DN	ou=People,dc=example,dc=com
Scope	sub
Filter	(isMemberOf=cn=Test Group,ou=Groups,dc=example,dc=com)
Requested Attributes	cn, mail

The `isDirectMemberOf` virtual attribute can be used in the examples above in place of `isMemberOf` if you only need to find groups that users are an actual member of. You must use `isMemberOf` for nested group membership.

Note that if this filter targets a dynamic group using an unindexed search, then this may be an expensive operation. However, it will not be any more expensive than retrieving the target group and then issuing a search based on information contained in the member URL.

For static groups, this approach has the added benefit of using a single search to retrieve information from all user entries, whereas it would otherwise be required to retrieve the static group and then perform a separate search for each member's entry.

Using static groups

A static group contains an explicit membership list where each member is represented as a DN-valued attribute. There are three types of static groups supported for use in the Directory Server:

- **groupOfNames**. A static group that is defined with the `groupOfNames` structural object class and uses the `member` attribute to hold the DNs of its members. RFC 4519 requires that the `member` attribute be required in an entry. However, the Directory Server has relaxed this restriction by making the `member` attribute optional so that the last member in the group can be removed. The following entry depicts a group defined with the `groupOfNames` object class:

  ```
  dn: cn=Test Group,ou=Groups,dc=example,dc=com
  objectClass: top
  objectClass: groupOfNames
  cn: Test Group
  member: uid=user.1,ou=People,dc=example,dc=com
  member: uid=user.2,ou=People,dc=example,dc=com
  member: uid=user.3,ou=People,dc=example,dc=com
  ```
- **groupOfUniqueNames**. A static group that is defined with the `groupOfUniqueNames` structural object class and uses the `uniquemember` attribute to hold the DNs of its members. RFC 4519 requires that the `uniquemember` attribute be required in an entry. However, the Directory Server has relaxed this restriction by making the `uniquemember` attribute optional so that the last member in the group can be removed. The following entry depicts a group defined with the `groupOfUniqueNames` object class:

  ```
  dn: cn=Test Group,ou=Groups,dc=example,dc=com
  objectClass: top
  objectClass: groupOfUniqueNames
  cn: Test Group
  uniquemember: uid=user.1,ou=People,dc=example,dc=com
  uniquemember: uid=user.2,ou=People,dc=example,dc=com
  uniquemember: uid=user.3,ou=People,dc=example,dc=com
  ```
- **groupOfEntries**. A static group that is defined with the `groupOfEntries` object class and uses the `member` attribute to hold the DNs of its members. This group specifies that the `member` attribute is optional to ensure that the last member can be removed from the group. Although the draft proposal (draft-findlay-ldap-groupofentries-00.txt) has expired, the Directory Server supports this implementation. The following entry depicts a group defined with the `groupOfEntries` object class:

  ```
  dn: cn=Test Group,ou=Groups,dc=example,dc=com
  objectClass: top
  objectClass: groupOfEntries
  cn: Test Group
  ```

```
member: uid=user.1,ou=People,dc=example,dc=com
member: uid=user.2,ou=People,dc=example,dc=com
member: uid=user.3,ou=People,dc=example,dc=com
```

Creating static groups

You can configure a static group by adding it using an LDIF file. Static groups contain a membership list of explicit DNs specified by the `uniquemember` attribute.

Creating a static group

Steps

1. Open a text editor, and then create a group entry in LDIF. Make sure to include the `groupOfUniquenames` object class and `uniquemember` attributes. If you did not have `ou=groups` set up in your server, then you can add it in the same file. When done, save the file as `static-group.ldif`. The following example LDIF file creates two groups, `cn=Development` and `cn=QA`.

   ```
   dn: ou=groups,dc=example,dc=com
   objectclass: top
   objectclass: organizationalunit
   ou: groups

   dn: cn=Development,ou=groups,dc=example,dc=com
   objectclass: top
   objectclass: groupOfUniqueNames
   cn: Development
   ou: groups
   uniquemember: uid=user.14,ou=People,dc=example,dc=com
   uniquemember: uid=user.91,ou=People,dc=example,dc=com
   uniquemember: uid=user.180,ou=People,dc=example,dc=com

   dn: cn=QA,ou=groups,dc=example,dc=com
   objectclass: top
   objectclass: groupOfUniqueNames
   cn: QA
   ou: groups
   uniquemember: uid=user.0,ou=People,dc=example,dc=com
   uniquemember: uid=user.1,ou=People,dc=example,dc=com
   uniquemember: uid=user.2,ou=People,dc=example,dc=com
   ```

2. Use **`ldapmodify`** to add the group entries to the server.

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

3. Verify the configuration by using the virtual attribute `isDirectMemberOf` that checks membership for a non-nested group. By default, the virtual attribute is disabled by default, but you can enable it using **`dsconfig`**.

   ```
   $ bin/dsconfig set-virtual-attribute-prop --name isDirectMemberOf --set
    enabled:true
   ```

4. Use **`ldapsearch`** to specifically search the `isDirectMemberOf` virtual attribute to determine if `uid=user.14` is a member of the `cn=Development` group. In this example, assume that administrator has the privilege to view operational attributes.

   ```
   $ bin/ldapsearch --baseDN dc=example,dc=com "(uid=user.14)" isDirectMemberOf

   dn: uid=user.14,ou=People,dc=example,dc=com
   isDirectMemberOf: cn=Development,ou=groups,dc=example,dc=com
   ```

5. Typically, you would want to use the group as a target in access control instructions. Open a text editor, create an `aci` attribute in an LDIF file, and save the file as `dev-group-aci.ldif`. Add the file

using the **ldapmodify** tool. You can create a similar ACI for the QA group, which is not shown in this example.

```
dn: ou=People,dc=example,dc=com
changetype: modify
add: aci
aci: (target ="ldap:///ou=People,dc=example,dc=com")
  (targetattr != "cn || sn || uid")
  (targetfilter ="(ou=Development)")
  (version 3.0; acl "Dev Group Permissions";
    allow (write) (groupdn = "ldap:///
cn=Development,ou=groups,dc=example,dc=com");)
```

6. Add the file using the **ldapmodify** tool.

```
$ bin/ldapmodify --filename dev-group-aci.ldif
```

Adding a new member to a static group

Steps

- To add a new member to the group, add a new value for the `uniquemember` attribute that specifies the DN of the user to be added. The following example adds a new `uniquemember`, `user.4`:

```
dn: cn=QA,ou=Groups,dc=example,dc=com
changetype: modify
add: uniquemember
uniquemember: uid=user.4,ou=People,dc=example,dc=com
```

Removing a member from a static group

Steps

- To remove a member from a static group, remove that user's DN from the `uniquemember` attribute. The following example removes the DN of `user.1`:

```
dn: cn=QA,ou=Groups,dc=example,dc=com
changetype: modify
delete: uniquemember
uniquemember: uid=user.1,ou=People,dc=example,dc=com
```

Searching static groups

The following sections describe how to compose searches to determine if a user is a member of a static group, to determine all the static groups in which a user is a member, and to determine all the members of a static group.

Determining if a user is a static group member

About this task

To determine whether a user is a member of a specified group, perform a base-level search to retrieve the group entry with an equality filter looking for the membership attribute of a value equal to the DN of the specified user.

For best performance, you will want to include a specific attribute list (just "cn", or "1.1" request that no attributes be returned) so that the entire member list is not returned. For example, to determine whether the user "`uid=john.doe,ou=People,dc=example,dc=com`" is a member of the `groupOfNames` static group "`cn=Test Group,ou=Groups,dc=example,dc=com`", issue a search with the following criteria:

Search Criteria for a Single User's Membership in a Static Group

Base DN	cn=Test Group,ou=Groups,dc=example,dc=com
Scope	base
Filter	(member=uid=john.doe,ou=People,dc=example,dc=com)
Requested Attributes	1.1

If the search returns an entry, then the user is a member of the specified group. If the search does not return any entries, then the user is not a member of the group. If you do not know the membership attribute for the specified group (it could be either a `member` or `uniqueMember` attribute), then you may want to revise the filter so that it allows either one as follows:

```
(|(member=uid=john.doe,ou=People,dc=example,dc=com)
(uniqueMember=uid=john.doe,ou=People,dc=example,dc=com))
```

Steps

- Run a base-level search to retrieve the group entry with an equality filter looking for the membership attribute.

```
$ bin/ldapsearch --baseDN "cn=Test Group,ou=Groups,dc=example,dc=com"
  --searchScope base "(member=uid=john.doe,ou=People,dc=example,dc=com)"
 "1.1"
```

Determining the static groups to which a user belongs

About this task

To determine the set of all static groups in which a user is specified as a member, perform a subtree search based at the top of the DIT. The search filter must be configured to match any type of static group in which the specified user is a member.

For example, the following criteria may be used to determine the set of all static groups in which the user, `uid=john.doc,ou=People,dc=example,dc=com`, is a member:

Search Criteria for Determining All the Static Groups for a User

Base DN	dc=example,dc=com
Scope	sub
Filter	(|(&(objectClass=groupOfNames) (member=uid=john.doe,ou=People,dc=example,dc=com)) (&(objectClass=groupOfUniqueNames)(uniqueMem- ber=uid=john.doe,ou=People,dc=example,dc=com)) (&(objectClass=groupOfEntries) (member=uid=john.doe,ou=People,dc=example,dc=com)))
Requested Attributes	1.1

Every entry returned from the search represents a static group in which the specified user is a member.

Steps

- Run a sub-level search to retrieve the static groups to which a user belongs.

```
$ bin/ldapsearch --baseDN "dc=example,dc=com" --searchScope sub \
  "(|(&(objectClass=groupOfNames)
```

```
  (member=uid=john.doe,ou=People,dc=example,dc=com)) \
  (&(objectClass=groupOfUniqueNames)\
  (uniqueMember=uid=john.doe,ou=People,dc=example,dc=com)) \
  (&(objectClass=groupOfEntries) \
  (member=uid=john.doe,ou=People,dc=example,dc=com)))" "1.1"
```

ⓘ **Note:** A base level search of the user's entry for `isMemberOf` or `isDirectMemberOf` virtual attributes will give the same results. You can also use the virtual attributes with virtual static groups.

Determining the members of a static group

About this task

To determine all of the members for a static group, simply retrieve the group entry including the membership attribute. The returned entry will include the DNs of all users that are members of that group. For example, the following criteria may be used to retrieve the list of all members for the group `cn=Test Group,ou=Groups,dc=example,dc=com`:

Search Criteria for All of the Static Group's Members

Base DN	cn=Test Group,ou=Groups,dc=example,dc=com
Scope	base
Filter	(objectClass=*)
Requested Attributes	member uniqueMember

If you want to retrieve additional information about the members, such as attributes from member entries, you must issue a separate search for each member to retrieve the user entry and the desired attributes.

Steps

- Run a base-level search to retrieve all of the members in a static group.

```
$ bin/ldapsearch --baseDN "cn=Test Group,ou=Groups,dc=example,dc=com" \
  --searchScope base "(objectclass=*)" uniqueMember
```

ⓘ **Note:** If you want to retrieve attributes from member entries, it is more efficient to search all users whose `isMemberOf` attribute contains the group DN, returning the attributes desired.

Using dynamic groups

Dynamic groups contain a set of criteria used to identify members rather than maintaining an explicit list of group members. If a new user entry is created or if an existing entry is modified so that it matches the membership criteria, then the user will be considered a member of the dynamic group. Similarly, if a member's entry is deleted or if it is modified so that it no longer matches the group criteria, then the user will no longer be considered a member of the dynamic group.

In the Directory Server, dynamic groups include the `groupOfURLs` structural object class and use the `memberurl` attribute to provide an LDAP URL that defines the membership criteria. The base, scope, and filter of the LDAP URL will be used in the process of making the determination, and any other elements present in the URL will be ignored. For example, the following entry defines a dynamic group in which all users below `dc=example,dc=com` with an `employeeType` value of contractor will be considered members of the group:

```
dn: cn=Sales Group,ou=groups,dc=example,dc=com
objectClass: top
objectClass: groupOfURLs
```

```
cn: Sales Group
memberURL: ldap:///dc=example,dc=com??sub?(employeeType=contractor)
```

Assuming that less than 80,000 entries have the `employeeType` of contractor, you need to create the following index definition to evaluate the dynamic group:

```
$ bin/dsconfig create-local-db-index --backend-name userRoot \
  --index-name employeeType --set index-entry-limit:80000 \
  --set index-type:equality
```

Creating dynamic groups

About this task

You can configure a dynamic group in the same manner as static groups using an LDIF file. Dynamic groups contain a membership list of attributes determined by search filter using an LDAP URL. You must use the `groupOfURLs` object class and the `memberURL` attribute.

To create a dynamic group:

Steps

1. Assume that `uid=user.15` is not part of any group. Use **`ldapsearch`** to verify that `uid=user.15` is not part of any group. In a later step, we will add the user to the dynamic group.

   ```
   $ bin/ldapsearch --baseDN dc=example,dc=com --searchScope sub
    "(uid=user.15)" ou
   ```

   ```
   dn: uid=user.15,ou=People,dc=example,dc=com
   ```

2. Assume for this example that `uid=user.0` has an `ou=Engineering` attribute indicating that he or she is a member of the Engineering department.

   ```
   $ bin/ldapsearch --baseDN dc=example,dc=com --searchScope sub "(uid=user.0)"
    ou isMemberOf
   ```

   ```
   dn: uid=user.0,ou=People,dc=example,dc=com
   ou: Engineering
   ```

3. Open a text editor, and then create a dynamic group entry in LDIF. The LDIF defines the dynamic group to include all users who have the `ou=Engineering` attribute. When done, save the file as `add-dynamic-group.ldif`.

   ```
   dn: cn=eng-staff,ou=groups,dc=example,dc=com
   objectclass: top
   objectclass: groupOfURLs
   ou: groups
   cn: eng-staff
   memberURL: ldap:///ou=People,dc=example,dc=com??sub?(ou=Engineering)
   ```

4. Use **`ldapmodify`** to add the group entry to the server.

   ```
   $ bin/ldapmodify --defaultAdd --filename add-dynamic-group.ldif
   ```

5. Use **`ldapsearch`** to specifically search the `isMemberOf` virtual attribute to determine if `uid=user.0` is a member of the `cn=Engineering` group or any other group.

   ```
   $ bin/ldapsearch --baseDN dc=example,dc=com "(uid=user.0)" isMemberOf
   ```

   ```
   dn: uid=user.0,ou=People,dc=example,dc=com
   isMemberOf: cn=eng-staff,ou=groups,dc=example,dc=com
   ```

6. If your data is relatively small (under 1 million entries), you can search for all users in the group that meet the search criteria (`ou=Engineering`). For very large databases, it is not practical to run a

database-wide search for all users as there can be a performance hit on the Directory Server. The following command returns the DNs of entries that are part of the `cn=eng-staff` dynamic group and sorts them in ascending order by the `sn` attribute.

```
$ bin/ldapsearch --baseDN dc=example,dc=com --sortOrder sn \
  "(isMemberOf=cn=eng-staff,ou=groups,dc=example,dc=com)" dn
```

7. Add `uid=user.15` to the `eng-staff` group by adding an `ou=Engineering` attribute to the entry. This step highlights an advantage of dynamic groups: you can make a change in an entry without explicitly adding the DN to the group as you would with static groups. The entry will be automatically added to the `eng-staff` dynamic group.

```
$ bin/ldapmodify
dn: uid=user.15,ou=People,dc=example,dc=com
changetype: modify
add: ou
ou: Engineering
```

8. Use **`ldapsearch`** to check if the user is part of the `cn=eng-staff` dynamic group.

```
$ bin/ldapsearch --baseDN dc=example,dc=com --searchScope sub
 "(uid=user.15)" isMemberOf
```

```
dn: uid=user.15,ou=People,dc=example,dc=com
isMemberOf: cn=eng-staff,ou=groups,dc=example,dc=com
```

Searching dynamic groups

The following sections describe how to compose searches to determine if a user is a member of a dynamic group, to determine all the dynamic groups in which a user is a member, and to determine all the members of a dynamic group.

Determining if a user is a dynamic group member

About this task

To determine whether a user is a member of a specific dynamic group, you must verify that the user's entry is both within the scope of the member URL and that it matches the filter contained in that URL. You can verify that a user's entry is within the scope of the URL using simple client-side only processing. Evaluating the filter against the entry on the client side can be more complicated. While possible, particularly in clients that are able to perform schema-aware evaluation, a simple alternative is to perform a base-level search to retrieve the user's entry with the filter contained in the member URL.

For example, to determine whether the user `uid=john.doe,ou=People,dc=example,dc=com` is a member of the dynamic group with the above member URL, issue a search with the following criteria:

Search Criteria for a Single User's Membership in a Dynamic Group

Base DN	uid=john.doe,ou=People,dc=example,dc=com
Scope	base
Filter	(ou=Engineering)
Requested Attributes	1.1

Note that the search requires the user DN to be under the search base defined in the `memberurl` attribute for the user to be a member. If the search returns an entry, then the user is a member of the specified group. If the search does not return any entries, then the user is not a member of the group.

Determining the dynamic groups to which a user belongs

About this task

To determine the set of all dynamic groups in which a user is a member, first perform a search to find all dynamic group entries defined in the server. You can do this using a subtree search with a filter of "`(objectClass=groupOfURLs)`".

You should retrieve the `memberURL` attribute so that you can use the logic described in the previous section to determine whether the specified user is a member of each of those groups. For example, to find the set of all dynamic groups defined in the `dc=example,dc=com` tree, issue a search with the following criteria:

Search Criteria for Determining All of the Dynamic Groups for a User

Base DN	dc=example,dc=com
Scope	sub
Filter	(objectClass=groupOfURLs)
Requested Attributes	memberURL

Each entry returned will be a dynamic group definition. You can use the base, scope, and filter of its `memberURL` attribute to determine whether the user is a member of that dynamic group.

Determining the members of a dynamic group

About this task

To determine all members of a dynamic group, issue a search using the base, scope, and filter of the member URL. The set of requested attributes should reflect the attributes desired from the member user entries, or "1.1" if no attributes are needed.

For example, to retrieve the **`cn`** and **`mail`** attributes from the group described above, use the following search:

Search Criteria for Determining the Members of a Dynamic Group

Base DN	dc=example,dc=com
Scope	sub
Filter	(employeeType=contractor)
Requested Attributes	cn, mail

ⓘ **CAUTION:** Note that this search may be expensive if the associated filter is not indexed or if the group contains a large number of members.

Using dynamic groups for internal operations

You can use dynamic groups for internal operations, such as ACI or component evaluation. The Directory Server performs the `memberurl` parsing and internal LDAP search; however, the internal search operation may not be performed with access control rules applied to it.

For example, the following dynamic group represents an organization's employees within the same department:

```
dn: cn=department 202,ou=groups,dc=example,dc=com
objectClass: top
```

```
objectClass: groupOfURLs
cn: department 202
owner: uid=user.1,ou=people,dc=example,dc=com
owner: uid=user.2,ou=people,dc=example,dc=com
memberURL: ldap:///ou=People,dc=example,dc=com??sub?
  (&(employeeType=employee)(departmentNumber=202))
description: Group of employees in department 202
```

The above group could be referenced from within the ACI at the `dc=example,dc=com` entry. For example:

```
dn:dc=example,dc=com
aci: (targetattr="employeeType")
  (version 3.0; acl "Grant write access to employeeType" ;
    allow (all) groupdn="ldap:///cn=department
 202,ou=groups,dc=example,dc=com";)
```

Any user matching the filter can bind to the server with their entry and modify the `employeeType` attribute within any entry under `dc=example,dc=com`.

Using virtual static groups

Static groups can be easier to interact with than dynamic groups, but large static groups can be expensive to manage and require a large amount of memory to hold in the internal group cache. The Directory Server provides a third type of group that makes it possible to get the efficiency and ease of management of a dynamic group while allowing clients to interact with it as a static group. A *virtual static group* is a type of group that references another group and provides access to the members of that group as if it was a static group.

To create a virtual static group, create an entry that has a structural object class of either `groupOfNames` or `groupOfUniqueNames` and an auxiliary class of `ds-virtual-static-group`. It should also include a `ds-target-group-dn` attribute, whose value is the group from which the virtual static group should obtain its members. For example, the following will create a virtual static group that exposes the members of the `cn=Sales Group,ou=Groups,dc=example,dc=com` dynamic group as if it were a static group:

```
dn: cn=Virtual Static Sales Group,ou=Groups,dc=example,dc=com
objectClass: top
objectClass: groupOfNames
objectClass: ds-virtual-static-group
cn: Virtual Static Sales Group
ds-target-group-dn: cn=Sales Group,ou=Groups,dc=example,dc=com
```

Note that you must also enable a virtual attribute that allows the `member` attribute to be generated based on membership for the target group. A configuration object for this virtual attribute does exist in the server configuration, but is disabled by default. To enable it, issue the following change:

```
$ bin/dsconfig set-virtual-attribute-prop --name "Virtual Static member" \
  --set enabled:true
```

If you want to use virtual static groups with the `groupOfUniqueNames` object class, then you will also need to enable the `Virtual Static uniqueMember` virtual attribute in the same way.

Creating virtual static groups

About this task

If your application only supports static groups but has scalability issues, then using a virtual static group could be a possible solution. A virtual static group uses a virtual attribute that is dynamically generated when called after which the operations that determine group membership are passed to another group, such as a dynamic group. You must use the `ds-virtual-static-group` object class and the `ds-target-group-dn` virtual attribute.

Virtual static groups are best used when determining if a single user is a member of a group. It is not a good solution if an application accesses the full list of group members due to the performance expense at constructing the list. If you have a small database and an application that requires that the full membership list be returned, you must also enable the `allow-retrieving-membership` property for the Virtual Static `uniqueMember` virtual attribute using the **`dsconfig`** tool.

To create a virtual static group:

Steps

1. Open a text editor, and then create a group entry in LDIF. The entry contains the `groupOfUniqueNames` object class, but in place of the `uniquemember` attribute is the `ds-target-group-dn` virtual attribute, which is part of the `ds-virtual-static-group` auxiliary object class. When done, save the file as `add-virtual-static-group.ldif`.

```
dn: cn=virtualstatic,ou=groups,dc=example,dc=com
objectclass: top
objectclass: groupOfUniqueNames
objectclass: ds-virtual-static-group
ou: groups
cn: virtual static
ds-target-group-dn: cn=eng-staff,ou=groups,dc=example,dc=com
```

2. Use **`ldapmodify`** to add the virtual static group entry to the server.

```
$ bin/ldapmodify -h server1.example.com -p 389 -D
 "uid=admin,dc=example,dc=com" \
  -w password -a -f add-virtual-static-group.ldif
```

3. Use **`dsconfig`** to enable the Virtual Static `uniqueMember` attribute, which is disabled by default.

```
$ bin/dsconfig set-virtual-attribute-prop --name "Virtual Static
 uniqueMember" \
  --set enabled:true
```

4. In the previous section, we set up `uid=user.0` to be part of the `cn=eng-staff` dynamic group. Use **`ldapsearch`** with the `isMemberOf` virtual attribute to determine if `uid=user.0` is part of the virtual static group.

```
$ bin/ldapsearch -h server1.example.com -p 389 -D "cn=Directory Manager" \
  -w secret -b dc=example,dc=com" "(uid=user.0)" isMemberOf
```

```
dn: uid=user.0,ou=People,dc=example,dc=com
isMemberOf: cn=virtualstatic,ou=groups,dc=example,dc=com
isMemberOf: cn=eng-staff,ou=groups,dc=example,dc=com
```

5. Use **`ldapsearch`** to determine if `uid=user.0` is a member of the virtual static group. You should see the returned `cn=virtualstatic` entry if successful.

```
$ ldapsearch -h localhost -p 1389 -D "cn=Directory Manager" -w password \
  -b "cn=virtualStatic,ou=Groups,dc=example,dc=com" \
  "(&(objectclass=groupOfUniqueNames) \
  (uniquemember=uid=user.0,ou=People,dc=example,dc=com))"
```

6. Next, try searching for a user that is not part of the `cn=eng-staff` dynamic group (e.g., `uid=user.20`), nothing will be returned.

```
$ ldapsearch -h localhost -p 1389 -D "cn=Directory Manager" -w password \
  -b "cn=virtualStatic,ou=Groups,dc=example,dc=com" \
  "(&(objectclass=groupOfUniqueNames) \
  (uniquemember=uid=user.20,ou=People,dc=example,dc=com))"
```

Searching virtual static groups

Because virtual static groups behave like static groups, the process for determining whether a user is a member of a virtual static group is identical to that of a member in a static group. Similarly, the process for determining all virtual static groups in which a user is a member is basically the same as the process as that of real static groups in which a user is a member. In fact, the query provided in the static groups discussion returns virtual static groups in addition to real static groups, because the structural object class of a virtual static group is the same as the structural object class for a static group.

You can also retrieve a list of all members of a virtual static group in the same way as a real static group: simply retrieve the `member` or `uniqueMember` attribute of the desired group. However, because virtual static groups are backed by dynamic groups and the process for retrieving member information for dynamic groups can be expensive, virtual static groups do not allow retrieving the full set of members by default. The virtual attribute used to expose membership can be updated to allow this with a configuration change such as the following:

```
$ bin/dsconfig set-virtual-attribute-prop --name "Virtual Static member" \
  --set allow-retrieving-membership:true
```

Because this can be an expensive operation, we recommend that the option to allow retrieving virtual static group membership be left disabled unless it is required.

Creating nested groups

About this task

The PingDirectory Server supports nested groups, where the DN of an entry that defines a group is included as a member in the parent entry. For example, the following example shows a nested static group (e.g., `cn=Engineering Group`) that has `uniquemember` attributes consisting of other groups, such as `cn=Developers Group` and the `cn=QA Group` respectively.

```
dn: cn=Engineering Group,ou=Groups,dc=example,dc=com
objectclass: top
objectclass: groupOfUniqueNames
cn: Engineering Group
uniquemember: cn=Developers,ou=Groups,dc=example,dc=com
uniquemember: cn=QA,ou=Groups,dc=example,dc=com
```

Nested group support is enabled by default on the Directory Server. To support nested groups without the performance hit, the Directory Server uses a group cache, which is also enabled by default. The cache supports static group nesting that includes other static, virtual static, and dynamic groups. The Directory Server provides a new monitoring entry for the group cache, `cn=Group Cache,cn=Monitor`.

In practice, nested groups are not commonly used for several reasons. LDAP specifications do not directly address the concept of nested groups, and some servers do not provide any level of support for them. Supporting nested groups in LDAP clients is not trivial, and many directory server-enabled applications that can interact with groups do not provide any support for nesting. If nesting support is not needed in your environment, or if nesting support is only required for clients but is not needed for server-side evaluation (such as for groups used in access control rules, criteria, virtual attributes, or other ways that the server may need to make a membership determination), then this support should be disabled.

To create nested static groups:

Steps

1. The following example shows how to set up a nested static group, which is a static group that contains `uniquemember` attributes whose values contain other groups (static, virtual static, or dynamic). Open a text editor, and then create a group entry in LDIF. Make sure to include the `groupOfUniquenames` object class and `uniquemember` attributes. If you did not have `ou=groups` set up in your server, then

you can add it in the same file. When done, save the file as `nested-group.ldif`. Assume that the static groups, `cn=Developers Group` and `cn=QA Group`, have been configured.

```
dn: ou=groups,dc=example,dc=com
objectclass: top
objectclass: organizationalunit
ou: groups

dn: cn=Engineering Group,ou=groups,dc=example,dc=com
objectclass: top
objectclass: groupOfUniqueNames
cn: Engineering Group
uniquemember: cn=Developers,ou=groups,dc=example,dc=com
uniquemember: cn=QA,ou=groups,dc=example,dc=com
```

2. Use **`ldapmodify`** to add the group entry.

```
$ bin/ldapmodify --defaultAdd --filename nested-static-group.ldif
```

3. Verify the configuration by using the `isMemberOf` virtual attribute that checks the group membership for an entry. By default, the virtual attribute is enabled. Use **`ldapsearch`** to specifically search the `isMemberOf` virtual attribute to determine if `uid=user.14` is a member of the `cn=Development` group. In this example, assume that the administrator has the privilege to view operational attributes.

```
$ bin/ldapsearch --baseDN dc=example,dc=com "(uid=user.14)" isMemberOf

dn: uid=user.14,ou=People,dc=example,dc=com
isMemberOf: cn=Development,ou=groups,dc=example,dc=com
```

4. Typically, you would want to use the group as a target in access control instructions. Open a text editor, create an ACI in LDIF, and save the file as `eng-group-aci.ldif`.

```
dn: ou=People,dc=example,dc=com
changetype: modify
add: aci
aci: (target ="ldap:///ou=People,dc=example,dc=com")
  (targetattr != "cn || sn || uid")
  (targetfilter ="(ou=Engineering Group)")
  (version 3.0; acl "Engineering Group Permissions";
    allow (write) (groupdn = "ldap:///cn=Engineering
 Group,ou=groups,dc=example,dc=com");)
```

5. Add the file using the **`ldapmodify`** tool.

```
$ bin/ldapmodify --filename eng-group-aci.ldif
```

ⓘ **Note:** When nesting dynamic groups, you cannot include other groups as members of a dynamic group. You can only support "nesting" by including the members of another group with a filter in the member URL. For example, if you have two groups `cn=dynamic1` and `cn=dynamic2`, you can nest one group in another by specifying it in the member URL as follows:

```
cn=dynamic1,ou=groups,dc=example,dc=com
objectClass: top
objectClass: groupOfURLs
memberURL: ldap:///dc=example,dc=com??sub?
(isMemberOf=cn=dynamic2,ou=groups,dc=example,dc=com)
```

The members included from the other group using this method are not considered "nested" members and will be returned even when using `isDirectMemberOf` when retrieving the members.

Maintaining referential integrity with static groups

The Directory Server can automatically update references to an entry whenever that entry is removed or renamed in a process called *referential integrity*. For example, if a user entry is deleted, then referential integrity plugin will remove that user from any static groups in which the user was a member (this is not necessary for dynamic groups, since no explicit membership is maintained). Similarly, if a modify DN operation is performed to move or rename a user entry, then referential integrity updates static groups in which that user is a member with the new user DN.

Referential integrity support is disabled by default, but may be enabled using the **`dsconfig`** tool as follows:

```
$ bin/dsconfig set-plugin-prop --plugin-name "Referential Integrity" \
  --set enabled:true
```

Other configuration attributes of note for this plugin include:

- **attribute-type**. This attribute specifies the names or OIDs of the attribute types for which referential integrity will be maintained. By default, referential integrity is maintained for the `member` and `uniqueMember` attributes. Any attribute types specified must have a syntax of either distinguished name (OID "1.3.6.1.4.1.1466.115.121.1.12") or name and optional UID (OID "1.3.6.1.4.1.1466.115.121.1.34"). The specified attribute types must also be indexed for equality in all backends for which referential integrity is to be maintained.
- **base-dn**. This attribute specifies the subtrees for which referential integrity will be maintained. If one or more values are provided, then referential integrity processing will only be performed for entries which exist within those portions of the DIT. If no values are provided (which is the default behavior), then entries within all public naming contexts will be included.
- **log-file**. This attribute specifies the path to a log file that may be used to hold information about the DNs of deleted or renamed entries. If the plugin is configured with a nonzero update interval, this log file helps ensure that appropriate referential integrity processing occurs even if the server is restarted.
- **update-interval**. This attribute specifies the maximum length of time that a background thread may sleep between checks of the referential integrity log file to determine whether any referential integrity processing is required. By default, this attribute has a value of "0 seconds", which indicates that all referential integrity processing is to be performed synchronously before a response is returned to the client. A duration greater than 0 seconds indicates that referential integrity processing will be performed in the background and will not delay the response to the client.

In the default configuration, where referential integrity processing is performed synchronously, the throughput and response time of delete and modify DN operations may be adversely impacted because the necessary cleanup work must be completed before the response to the original operation can be returned. Changing the configuration to use a non-zero update interval alleviates this performance impact because referential integrity processing uses a separate background thread and does not significantly delay the response to delete or modify DN operations.

However, performing referential integrity processing in a background thread may introduce a race condition that may adversely impact clients that delete a user and then immediately attempt to re-add it and establish new group memberships. If referential integrity processing has not yet been completed for the delete, then newly-established group memberships may be removed along with those that already existed for the previous user. Similarly, if the newly-created user is to be a member of one or more of the same groups as the previous user, then attempts by the client to re-establish those memberships may fail if referential integrity processing has not yet removed the previous membership. For this reason, we recommend that the default synchronous behavior be maintained unless the performance impact associated with it is unacceptable and clients are not expected to operate in a manner that may be adversely impacted by delayed referential integrity processing.

ⓘ **Note:** The internal operations of the referential integrity plugin are not replicated. So, in a replicated topology, you must enable the referential integrity plugin consistently on all servers in the topology to ensure that changes made by the referential integrity plugin are passed along to a replication server.

For more information about administering the referential integrity plugin, see Chapter 6, “Configuring the Directory Server” in the *PingDirectory Server Administration Guide*.

Monitoring the group membership cache

The Directory Server logs information at startup about the memory consumed by the group membership cache. This hard-coded cache contains information about all of the group memberships for internal processing, such as ACIs. The group membership cache is enabled by default.

The information about this cache is logged to the standard output log (`server.out`) and the standard error log. When using groups, you can use the log information to tune the server for best performance. For example, at startup the server logs a message like the following to the `server.out` log:

```
[16/Aug/2011:17:14:39.462 -0500] category=JEB severity=NOTICE msgID=1887895587
msg="The database cache now holds 3419MB of data and is 32 percent full"
```

The error log will contain something like the following:

```
[16/Aug/2011:18:40:39.555 -0500] category=EXTENSIONS severity=NOTICE
 msgID=1880555575
msg="'Group cache (174789 static group(s) with 7480151 total memberships and
 1000002
unique members, 0 virtual static group(s), 1 dynamic group(s))' currently
 consumes
149433592 bytes and can grow to a maximum of 149433592 bytes"
```

Using the entry cache to improve the performance of large static groups

The PingDirectory Server provides an entry cache implementation, which allows for fine-grained control over the kinds of entries that may be held in the cache. You can define filters to specify the entries included in or excluded from the cache, and you can restrict the cache so that it holds only entries with at least a specified number of values for a given set of attributes.

Under most circumstances, we recommend that the Directory Server be used *without* an entry cache. The Directory Server is designed to efficiently retrieve and decode entries from the database in most cases. The database cache is much more space-efficient than the entry cache, and heavy churn in the entry cache can adversely impact garbage collection behavior.

However, if the Directory Server contains very large static groups, such as those containing thousands or millions of members, and clients need to frequently retrieve or otherwise interact with these groups, then you may want to enable an entry cache that holds only large static groups.

In servers containing large static groups, you can define an entry cache to hold only those large static groups. This entry cache should have an include filter that matches only group entries (for example, "`(|(objectclass=groupOfNames)(objectclass=groupOfUniqueNames)(objectclass=groupOfEntries))`"). The filter contains a minimum value count so that only groups with a large number of members (such as those with at least 100 `member` or `uniqueMember` values) will be included. The Directory Server provides an entry cache implementation with these settings although it is disabled by default.

Enabling the entry cache

Steps

- Run **`dsconfig`** to enable the entry cache.

```
$ bin/dsconfig set-entry-cache-prop --cache-name "Static Group Entry Cache"
 \
  --set enabled:true
```

Creating your own entry cache for large groups

Steps

- You can create your own entry cache for large groups using the **`dsconfig create-entry-cache`** subcommand.

```
# bin/dsconfig create-entry-cache --type fifo \
  --set enabled:true \
  --set cache-level:10 \
  --set max-entries:175000 \
  --set "include-filter:(objectClass=groupOfUniqueNames)" \
  --set min-cache-entry-value-count:10000 \
  --set min-cache-entry-attribute:uniquemember
```

Monitoring the entry cache

You can monitor the memory consumed by your entry cache using the `entry-cache-info` property in the periodic stats logger. You can retrieve the monitor entry over LDAP by issuing a search on `baseDN="cn=monitor"` using `filter="(objectClass=ds-fifo-entry-cache-monitor-entry)"`. For example, the entry might appear as follows:

```
dn: cn=Static Group Entry Cache Monitor,cn=monitor
objectClass: top
objectClass: ds-monitor-entry
objectClass: ds-fifo-entry-cache-monitor-entry
objectClass: extensibleObject
cn: Static Group Entry Cache Monitor
cacheName: Static Group Entry Cache
entryCacheHits: 6416407
entryCacheTries: 43069073
entryCacheHitRatio: 14
maxEntryCacheSize: 12723879900
currentEntryCacheCount: 1
maxEntryCacheCount: 175000
entriesAddedOrUpdated: 1
evictionsDueToMaxMemory: 0
evictionsDueToMaxEntries: 0
entriesNotAddedAlreadyPresent: 0
entriesNotAddedDueToMaxMemory: 0
entriesNotAddedDueToFilter: 36652665
entriesNotAddedDueToEntrySmallness: 0
lowMemoryOccurrences: 0
percentFullMaxEntries: 0
jvmMemoryMaxPercentThreshold: 75
jvmMemoryCurrentPercentFull: 24
jvmMemoryBelowMaxMemoryPercent: 51
isFull: false
capacityDetails: NOT FULL: The JVM is using 24% of its available memory.
 Entries can be
added to the cache until the overall JVM memory usage reaches the configured
 limit of
75%. Cache has 174999 remaining entries before reaching the configured limit
 of 175000.
```

By default, the entry cache memory is set to 75%, with a maximum of 90%.

Tuning the index entry limit for large groups

The Directory Server uses indexes to improve database search performance and provide consistent search rates regardless of the number of database objects stored in the DIT. You can specify an index entry limit property, which defines the maximum number of entries that are allowed to match a given index

key before it is no longer maintained by the server. If the index keys have reached this limit (which is 4000 by default), then you must rebuild the indexes using the **rebuild-index** tool as follows:

```
$ bin/rebuild-index --baseDN dc=example,dc=com --index objectclass
```

In the majority of Directory Server environments, the default index entry limit value of 4000 entries should be sufficient. However, group-related processing, it may be necessary to increase the index entry limit. For directories containing more than 4000 groups with the same structural object class (i.e., more than 4000 entries, 4000 `groupOfUniqueNames` entries, 4000 `groupOfEntries` entries, or 4000 `groupOfURLs` entries), then you may want to increase the index entry limit for the `objectClass` attribute so that it has a value larger than the maximum number of group entries of each type. Set `index-entry-limit` property using a command line like the following:

```
$ bin/dsconfig set-local-db-index-prop --backend-name userRoot \
  --index-name objectClass --set index-entry-limit:175000
```

As an alternative, a separate backend may be created to hold these group entries, so that an unindexed search in that backend yields primarily group entries. If you make no changes, then the internal search performed at startup to identify all groups and any user searches looking for groups of a given type may be very expensive.

For directories in which any single user may be a member of more than 4000 static groups of the same type, you may need to increase the index entry limit for the `member` and/or `uniqueMember` attribute to a value larger than the maximum number of groups in which any user is a member. If you do not increase the limit, then searches to retrieve the set of all static groups in which the user is a member may be unindexed and therefore very expensive.

Summary of commands to search for group membership

The following summary of commands show the fastest way to retrieve direct or indirect member DNs for groups.

- To retrieve direct member (non-nested) DNs of group "`cn=group.1,ou=groups,dc=example,dc=com`".

  ```
  $ bin/ldapsearch --baseDN "cn=group.1,ou=Groups,dc=example,dc=com"
   "(objectClass=*)" uniqueMember member
  ```
- To retrieve direct member entries (non-nested) under "`dc=example,dc=com`" of group "`cn=group.1,ou=groups,dc=example,dc=com`". This is useful when attributes from member entries are used in the filter or being returned.

  ```
  $ bin/ldapsearch --baseDN "ou=people,dc=example,dc=com"
   "(isDirectMemberOf=cn=group.1,ou=Groups,dc=example,dc=com)"
  ```
- To retrieve group DNs in which user "`uid=user.2,ou=people,dc=example,dc=com`" is a direct member (non-nested, static groups).

  ```
  $ bin/ldapsearch --baseDN "uid=user.2,ou=people,dc=example,dc=com"
   "(objectClass=*)" isDirectMemberOf
  ```
- To retrieve all member entries under `ou=people,dc=example,dc=com` of group "`cn=group.1,ou=groups,dc=example,dc=com`".

  ```
  $ bin/ldapsearch --baseDN "ou=people,dc=example,dc=com"
   "(isMemberOf=cn=group.1,ou=Groups,dc=example,dc=com)"
  ```
- To retrieve the group DNs in which user "`uid=user.2,ou=people,dc=example,dc=com`" is a member.

  ```
  $ bin/ldapsearch --baseDN "uid=user.2,ou=people,dc=example,dc=com"
   "(objectClass=*)" isMemberOf
  ```

Migrating Sun/Oracle groups

You can migrate Sun/Oracle static and dynamic groups to PingDirectory Server groups. The following sections outline the procedures for migrating static groups to both Ping Identity static groups and virtual static groups as well as how to migrate dynamic groups. For information about the differences in access control evaluation between Sun/Oracle and the PingDirectory Server, see Migrating ACIs from Sun/Oracle to PingDirectory Server.

Migrating static groups

About this task

The PingDirectory Server supports static LDAP groups with structural object classes of `groupOfNames`, `groupOfUniqueNames`, or `groupOfEntries`. In general, static groups may be imported without modification.

A FIFO entry cache can be enabled to cache group-to-user mappings, which improves performance when accessing very large entries, though at the expense of greater memory consumption. The PingDirectory Server provides an out-of-the-box FIFO entry cache object for this purpose. This object must be explicitly enabled using **`dsconfig`** as described in *Using the Entry Cache to Improve the Performace of Large Static Groups*.

To migrate static groups:

Steps

1. Run the **`migrate-ldap-schema`** tool to enumerate any schema differences between the DSEE deployment and the Ping Identity deployment.
2. Run the **`migrate-sun-ds-config`** tool to enumerate any configuration differences between the DSEE deployment and the Ping Identity deployment.
3. Import or configure any necessary schema and/or configuration changes recorded by the above tools.
4. Import the existing users and groups using the **`import-ldif`** tool.
5. From the PingDirectory Server root directory, open the `sun-ds-compatibility.dsconfig` file in the `docs` folder using a text editor.
6. Find the **FIFO Entry Cache** section and, after reading the accompanying comments, enable the corresponding **`dsconfig`** command by removing the comment character ("#").

   ```
   $ bin/dsconfig set-entry-cache-prop \
     --cache-name "Static Group Entry Cache" --set enabled:true
   ```
7. To ensure that references to an entry are updated automatically when the entry is deleted or renamed, enable the Referential Integrity plugin.

   ```
   $ bin/dsconfig set-plugin-prop --plugin-name "Referential Integrity" --set
    enabled:true
   ```

   If this Directory Server is part of a replication topology, you should enable the Referential Integrity plugin for each replica.

Migrating static groups to virtual static groups

About this task

In many cases, electing to use virtual static groups in place of static groups can produce marked performance gains without any need to update client applications. The specifics of a migration to virtual static groups varies depending on the original DIT, but the general approach involves identifying common membership traits for all members of each group and then expressing those traits in the form of an LDAP URL.

In the following example, the common membership trait for all members of the All Users group is the parent DN ou=People,dc=example,dc=com. In other cases, a common attribute may need to be used. For example, groups based on the location of its members could use the `l` (location) or `st` (state) attribute.

In the following example, consider the common case of an "All Users" group, which contains all entries under the parent DN "`ou=People,dc=example,dc=com`". When implemented as a virtual static group, this group may have a large membership set without incurring the overhead of a static group.

To migrate dsee static groups to virtual static groups:

Steps

1. First, create a dynamic group.

```
dn: cn=Dynamic All Users,ou=Groups,dc=example,dc=com
objectClass: top
objectClass: groupOfURLs
cn: Dynamic All Users
memberURL: ldap:///ou=People,dc=example,dc=com??sub?(objectClass=person)
```

2. Next, create a virtual static group that references the dynamic group.

```
dn: cn=All Users,ou=Groups,dc=example,dc=com
objectClass: top
objectClass: groupOfUniqueNames
objectClass: ds-virtual-static-group
cn: All Users
ds-target-group-dn: cn=Dynamic All Users,ou=Groups,dc=example,dc=com
```

3. Finally, the `Virtual Static uniqueMember` virtual attribute must be enabled to populate the All Users group with `uniqueMember` virtual attributes.

```
$ bin/dsconfig set-virtual-attribute-prop --name "Virtual Static
 uniqueMember" \
  --set enabled:true
```

4. Confirm that the virtual static group is correctly configured by checking a user's membership in the group.

```
$ bin/ldapsearch --baseDN "cn=All Users,ou=Groups,dc=example,dc=com" \
  --searchScope base "(uniqueMember=uid=user.0,ou=People,dc=example,dc=com)"
 1.1

dn: cn=All Users,ou=Groups,dc=example,dc=com
```

5. The ability to list all members of a virtual static group is disabled by default. You may enable this feature, but only if specifically required by a client application.

```
$ bin/dsconfig set-virtual-attribute-prop --name "Virtual Static
 uniqueMember" \
  --set allow-retrieving-membership: true
```

ⓘ **Note:** The virtual static group may also be implemented using the `groupOfNames` object class instead of `groupOfUniqueNames`. In that case, you must update the `Virtual Static member` configuration object instead of the `Virtual Static uniqueMember` configuration object.

Migrating dynamic groups

About this task

The PingDirectory Server supports dynamic groups with the `groupofURLs` object class. In general, dynamic groups may be imported without modification.

To migrate dynamic groups:

Steps

1. Run the **`migrate-ldap-schema`** tool to enumerate any schema differences between the DSEE deployment and the Ping Identity deployment.
2. Run the **`migrate-sun-ds-config`** tool to enumerate any configuration differences between the DSEE deployment and the Ping Identity deployment.
3. Import or configure any necessary schema and/or configuration changes recorded by the above tools.
4. Import the existing users and groups using the **`import-ldif`** tool.

Working with Indexes

The PingDirectory Server uses indexes to improve database search performance and provide consistent search rates regardless of the number of database objects stored in the Directory Information Tree (DIT). Indexes are associated with attributes and stored in database index files, which are managed separately for each base DN in the Directory Server.

Overview of indexes

The PingDirectory Server uses indexes to improve database search performance and provide consistent search rates regardless of the number of database objects stored in the Directory Information Tree (DIT). Indexes are associated with attributes and stored in database index files, which are managed separately for each base DN in the Directory Server. The Directory Server automatically creates index files when you first initialize a base DN or when you use the **`dsconfig`** tool to create a local DB backend. During modify operations, the Directory Server updates the database index files. If encryption is enabled on the server, indexes will be encrypted.

The PingDirectory Server comes with the following types of indexes:

- **Default system indexes** to ensure that the server operates efficiently. Indexes consist of database files that contain index keys mapping to the list of entry IDs.
- **Default Local DB indexes** that are created for each database suffix. Modify the index to meet your system's requirements using the **`dsconfig`** tool.
- **Local DB VLV indexes** that allow a client to request the server to send search results using the Virtual List View control.
- **Filtered Indexes** that provide the ability to index an attribute but only for entries that match a specified filter based on an equality index. The filtered index can only be used for searches containing that filter. The filtered index can be maintained independently of the equality index for that attribute and even if a normal equality index is not maintained for that attribute.

General tips on indexes

Administrators should keep the following tips in mind when working with indexes:

- **Important Critical Indexes**. The Directory Server has several built-in indexes on the Local DB Backend that are critical to internal server processing and should never be removed.

  ```
  aci, ds-entry-unique-id, objectClass
  ```

- **Built-in Indexes for Efficient Queries**. The Directory Server has built-in indexes on the Local DB Backend. Internal processing of the server relies on the `aci`, `ds-soft-delete-from-dn`, `ds-soft-delete-timestamp`, `entryUUID`, `member`, `objectClass`, and `uniqueMember` indexes, which must not be removed. The `mail` and `uid` indexes can be removed, but these attributes are referenced from the Password Modify Extended Operation and will cause problems with components such as the Exact Match Identity Mapper. If the `mail` or `uid` indexes are removed, additional configuration changes may be necessary to ensure that the server starts properly. The `cn`, `givenName`, `mail`, `sn`,

and `telephoneNumber` indexes can be safely removed if clients do not query on these attributes. This will reduce the size of the database both on disk and in memory.

- **Online Rebuilds**. Whenever an online index rebuild is in progress, the data in that backend will be available and writable although the index being rebuilt will not be used; therefore, searches which attempt to use that attribute might be unindexed.
- **Index Rebuild Administrative Alert**. The Directory Server generates an administrative alert when the rebuild process begins and ends. It will have a degraded-alert-type of "index-rebuild-in-progress" so that a Directory Proxy Server, such as the Directory Server can avoid using that server while the rebuild is in progress.
- **System Indexes Cannot be Rebuilt**. The contents of the backend must be exported and re-imported in order to rebuild system indexes. See the table below for the list of system indexes.
- **Indexing Certain Attributes**. You should ensure that the following recommendations are used when setting up the indexes.
  - Equality and substring indexes should not be used for attributes that contain binary data.
  - Approximate indexes should be avoided for attributes containing numbers, such as telephone numbers.
- **Unindexed Searches**. Unindexed attributes result in longer search times as the database itself has to be searched instead of the database index file. Only users with the `unindexed-search` privilege are allowed to carry out unindexed searches. In general, applications should be prevented from performing unindexed searches, so that searches that are not indexed would be rejected rather than tying up a worker thread. The ways to achieve this include:
  - Make sure that only the absolute minimum set of users have the `unindexed-search` privilege. This privilege can be used without any other restrictions.
  - To allow unindexed searches with some control, the Permit Unindexed Search request control can be used with the `unindexed-search-with-control` privilege. With this privilege, a user will only be permitted to request an unindexed search if the search request includes the Permit Unindexed Search request control. The `unindexed-search privilege` allows a client to request an unindexed search without this control.
  - The Reject Unindexed Search request control can be used to explicitly indicate that a client does not want the server to process an unindexed search request, regardless of priviledges. See the LDAP SDK for information about these controls. These capabilities are also available with the `ldapsearch` tool.
  - Make sure that `allow-unindexed-searches` property is set to false in all client connection policies, in which unindexed searches should never be necessary. If the client connection policy should allow undindexed searches, set the `allow-unindexed-searches-with-control` property to true. If `allow-unindexed-searches` is false but, `allow-unindexed-searches-with-control` is true, the policy will only permit an unindexed search if the request includes the Permit Unindexed Search request control. See the LDAP SDK and the `ldapsearch` tool for more information.
  - Set a nonzero value for the `maximum-concurrent-unindexed-searches` global configuration property to ensure that if unindexed searches are allowed, only a limited number of them will be active at any given time. Administrators can configure the maximum number of concurrent unindexed searches by setting a property under Global Configuration.

    To change the maximum number of concurrent unindexed searches, use the **`dsconfig`** tool to set a value for the number. A value of "0" (default) represents no limit on the number of concurrent unindexed searches.

    ```
    $ bin/dsconfig set-global-configuration-prop \
      --set maximum-concurrent-unindexed-searches:2
    ```
- **Index Entry Limit**. The Directory Server specifies an index entry limit property. This property defines the maximum number of entries that are allowed to match a given index key before it is no longer maintained by the server. If the index keys have reached this limit (default value is 4000), then you must

rebuild the indexes using the **`rebuild-index`** tool. If an index entry limit value is set for the local DB backend, it overrides the value set for the overall JE backend index entry limit configuration (i.e., 4000).

To change the default index entry limit, use the **`dsconfig`** tool as seen in the following example:

```
$ bin/dsconfig set-local-db-index-prop --backend-name userRoot \
  --index-name cn --set index-entry-limit:5000
```

- **Rebuild Index vs Full Import**. You can expect a limited amount of database growth due to the existence of old data when running **`rebuild-index`** versus doing a full import of your database.

Index types

The PingDirectory Server supports several types of indexes to quickly find entries that match search criteria in LDAP operations. The Directory Server uses an attribute's matching rules to normalize its values and uses those values as index keys to a list of matching entry IDs. Entry IDs are integer values that are used to uniquely identify an entry in the backend by means of a set of database index files (`id2entry`, `dn2id`, `dn2uri`, `id2children`, `id2subtree`).

Matching rules are elements defined in the schema that tell the server how to interact with the particular attribute. For example, the uid attribute has an equality matching rule defined in the schema, and thus, has an equality index maintained by the Directory Server. The following table describes the index types:

Directory Server Index Types

Index Type	Description
Approximate	Used to efficiently locate entries that match the approximate search filter. It is used to identify which entries are approximately equal to a given assertion. Approximate indexes can only be applied to attributes that have a corresponding approximate matching rule.
Equality	Used to efficiently locate entries that match the equality search filter. It is used to identify which entries are exactly equal to a given assertion. Equality indexes can only be applied to attributes that have a corresponding equality matching rule. An offshoot of the equality index is the filtered index, which uses a defined search filter for a specific attribute. The filtered index can be maintained independently of the equality index for a specific attribute.
Ordering	Used to efficiently locate entries that match the ordering search filter. It is used to identify which entries have a relative order of values for an attribute. Ordering indexes can only be applied to attributes that have a corresponding ordering matching rule.
Presence	Used to efficiently locate entries that match the presence search filter. It is used to identify which entries have at least one value for a specified attribute. There is only one presence index key per attribute.
Substring	Used to efficiently locate entries that match the substring search filter. It is used to identify which entries contain specific substrings to a given assertion. Substring indexes can only be applied to attributes that have a corresponding substring matching rule.

System indexes

The PingDirectory Server contains a set of system database index files that ensure that the server operates efficiently. These indexes cannot be modified or deleted.

System Indexes

Index	Description
dn2id	Allows quick retrieval of DNs. The DN database, or `dn2id`, has one record for each entry. The key is the normalized entry DN and the value is the entry ID.
id2entry	Allows quick retrieval of entries. The `id2entry` database contains the LDAP entries. The database key is the entry ID and the value is the entry contents.
referral	Allows quick retrieval of referrals. The referral database called `dn2uri` contains URIs from referral entries. The key is the DN of the referral entry and the value is that of a labeled URI in the ref attribute for that entry.
id2children	Allows quick retrieval of an entry and its children. The `id2children` database provides a mapping between an entry's unique identifier and the entry unique identifiers of the corresponding entry's children.
id2subtree	Allows quick retrieval of an entry's subtree. The `id2subtree` database provides a mapping between an entry's unique identifier and its unique identifiers in its subtree.

Viewing the system indexes

Steps

Use the **`dbtest`** command to view the system and user indexes. The index status indicates if it is a trusted index or not. An index is either trusted or untrusted. An untrusted index requires rebuilding.

```
$ bin/dbtest list-index-status --baseDN dc=example,dc=com --backendID userRoot
```

Managing local DB indexes

You can modify the local DB indexes to meet your system's requirements using the **`dsconfig`** tool. If you are using the **`dsconfig`** tool in interactive command-line mode, you can access the **Local DB Index** menu from the Basic object menu.

Viewing the list of local DB indexes

Steps

- Use **`dsconfig`** with the `list-local-db-indexes` option to view the default list of indexes.

```
$ bin/dsconfig list-local-db-indexes --backend-name userRoot
```

```
Local DB Index     : Type    : index-type
-------------------:---------:--------------------
aci                : generic : presence
cn                 : generic : equality, substring
ds-entry-unique-id : generic : equality
givenName          : generic : equality, substring
mail               : generic : equality
member             : generic : equality
objectClass        : generic : equality
sn                 : generic : equality, substring
telephoneNumber    : generic : equality
uid                : generic : equality
uniqueMember       : generic : equality
```

Viewing a property for all local DB indexes

Steps

- Use **dsconfig** with the `--property` option to view a property assigned set for all local DB indexes. Repeat the option for each property that you want to list. In this example, the `prime-index` property specifies if the backend is configured to prime the index at startup.

```
$ bin/dsconfig list-local-db-indexes --property index-entry-limit \
  --property prime-index --backend-name userRoot
```

Viewing the configuration parameters for local DB index

Steps

- To view the configuration setting of a local DB index, use **dsconfig** with the `get-local-db-index-prop` option and the `--index-name` and `--backend-name` properties. If you want to view the advanced properties, add the `--advanced` option to your command.

```
$ bin/dsconfig get-local-db-index-prop --index-name aci \
  --backend-name userRoot
```

Modifying the configuration of a local DB index

About this task

You can easily modify an index using the **dsconfig** tool. Any modification or addition of an index requires the indexes to be rebuilt. In general, an index only needs to be built once after it has been added to the configuration.

If you add an index, then import the data using the **import-ldif** tool, then the index will be automatically rebuilt. If you add an index, then add the data using some other method than **import-ldif**, you must rebuild the index using the **rebuild-index** tool.

Steps

1. Use **dsconfig** with the `set-local-db-index-prop` option and the `--index-name` and `--backend-name` properties. In this example, update the `prime-index` property, which loads the index at startup. This command requires the `--advanced` option to access this property.

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

2. View the index to verify the change.

```
$ bin/dsconfig get-local-db-index-prop --index-name uid \
  --backend-name userRoot
```

3. Stop the Directory Server. You can do an index rebuild with the server online; however, keep in mind the tips presented in *General Tips on Indexes*.

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

4. Run the **rebuild-index** tool.

```
$ bin/rebuild-index --baseDN dc=example,dc=com --index uid
```

5. Restart the Directory Server if shutdown.

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

Creating a new local DB index

Steps

1. To create a new local DB index, use **dsconfig** with the `--create-local-db-index` option and the `--index-name`, `--backend-name`, and `--set index-type: <value>` options.

```
$ bin/dsconfig create-local-db-index \
  --index-name roomNumber --backend-name userRoot \
  --set index-type:equality
```

2. View the index.

```
$ bin/dsconfig get-local-db-index-prop \
  --index-name roomNumber --backend-name userRoot
```

3. Stop the Directory Server. You can do an index rebuild with the server online; however, keep in mind the tips presented in *General Tips on Indexes* on page 148.

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

4. Rebuild the index using the **rebuild-index** tool.

```
$ bin/rebuild-index --baseDN dc=example,dc=com --index roomNumber
```

5. Restart the Directory Server.

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

Deleting a local DB index

About this task

You can delete an index using the **dsconfig** tool. Check that no plugin applications are using the index before deleting it. When the index is deleted, the corresponding index database will also be deleted. The disk space is reclaimed once the cleaner threads begin.

Steps

- Use **dsconfig** with the `delete-local-db-index` option to remove it from the database.

```
$ bin/dsconfig delete-local-db-index \
  --index-name roomNumber --backend-name userRoot
```

Working with composite indexes

The PingDirectory Server composite index can be generated from multiple pieces of information (a combination of multiple filter components, or a combination of filter components and a base DN). A composite index can also be based on only a single piece of information.

To improve searches over a large number of entries, equality composite indexes can be used to combine a mandatory equality filter pattern with an optional base DN pattern to improve the performance of searches in directories with a very large number of entries, and in particular with a very large number of non-leaf entries. Equality composite indexes offer two advantages over existing equality attribute indexes in these types of deployments.

Base DN Pattern - If a directory environment has many branches, but searches are often done that are within specific individual branches, the base DN pattern can be used to make search processing more efficient. The server will only need to search entries within a target branch.

For example, if the directory contains an **"ou=Customers,dc=example,dc=com"** branch, with a separate branches below that for sets of customers, like "ou=ACME,ou=Customers,dc=example,dc=com", and

"ou=SHOPCO,ou=Customers,dc=example,dc=com", a composite index with a filter pattern of **"(sn=?)"** and a base DN pattern of **"ou=?,ou=Customers,dc=example,dc=com"** can be defined. Then a search with a filter of **"(sn=Smith)"** and a base DN of **"ou=ACME,ou=Customers,dc=example,dc=com"** can be used to narrow the search to the Smiths in the ACME branch.

Index Pages - If many entries have the same value for a specific attribute, composite indexes can break large ID sets up across multiple pages, unlike the traditional attribute index. Using the previous example, if a search of the directory returns 50,000 Smiths, the results can be served in blocks of 5,000 IDs. An attribute index will return either one Smith record whose value is a block that contains all 50,000 of the matching entry IDs (if the index isn't exploded), or 50,000 Smith records that each have a value of the matching entry ID (when the index is exploded). The non-exploded form is efficient for searching because all of the entry IDs are returned in a single read, but it's expensive for writing because if a Smith must be added or removed, the entire block of 50,000 entry IDs must be rewritten. The exploded form is efficient for writing (adding or removing a Smith involves just that one entry ID), but it's expensive for searching because it takes 50,000 reads to get all entry IDs for all of the Smiths.

Composite indexes break up the block of entry IDs across multiple pages (a page size of up to 5000). If the directory contains 50,000 Smiths, instead of having to choose between one block of 50,000 IDs or 50,000 blocks of one ID, ten blocks of 5,000 IDs are returned. This improves the efficiency of a read or write across many entries.

There is little performance overhead to the paging mechanism. Use an equality composite index for an attribute that has a lot of entries that have the same value (such as givenName or sn), not for an attribute with very few entries with the same value (such as id or mail). For attributes in which all of the values match a small number of entries, it's better to use an equality attribute index.

When configuring a composite index, define the following properties:

Composite Index Properties

Composite Index Properties	Description
index-filter-pattern	Specifies a single-valued filter property used to identify a portion of the index criteria. This can only be specified at the time that the index definition is created and is required.
index-base-dn-pattern	Specifies a single-valued DN property that may indicate that the index should be scoped to a specific subtree or subtree pattern. This can only be specified at the time the index definition is created and is optional.

Working with JSON indexes

JSON indexing is similar to general attribute indexing. Where an attribute can be indexed several ways and requires a separate database for each index type, there is only a single database for each JSON field that can be used for different JSON filter types. This database primarily behaves like the database for an equality attribute index. Each database entry key is the normalized form for a value for the target JSON field, and the corresponding database entry value is an list of the entry IDs for all entries in which the associated attribute type has a JSON object with that value for the target field. The database is configured with a comparator (based on the data type for the target field) that enables iterating through values in a logical order to facilitate inequality and subInitial searches.

JSON indexes are automatically created when the JSON Field constraint indicates that a JSON field should be indexed. Indexes can be viewed with the following command:

```
$ bin/dbtest list-index-status \
  --backendID userRoot \
  --baseDN dc=example,dc=com
```

The JSON object filter types that can be enhanced through the use of JSON indexes include:

- `equals`. Identifies entries that have a specific value for the target field. This filter type only requires retrieving a single index key. However, depending on the nature of the search filter, the ID list may contain references to entries that don't actually match the filter (such as if the field is a string, and the filter is configured to use case-sensitive matching).
- `equalsAny`. Identifies entries that have any of a specified set of values for the target field. This filter type only requires retrieving the index keys that correspond to the target values in the filter and merging their ID lists.
- `greaterThan/lessThan`. Identifies entries that have at least one value for the target field that is greater or less than (or possibly equal to) a specified value. This index is similar in use to the `containsField` index, except that it only needs to iterate through a subset of the keys. A filter can contain both `greaterThan` and `lessThan` filters to represent a bounded range.
- `substring`. Identifies entries that have a string value for the target field that matches a given `substring`. The index can only be used for substring filters that include a `subInitial` component. In this case, the server iterates through all of the index keys that match the `startsWith` component, and manually compares values against the remainder of the substring assertion.

JSON indexing is available in local database backends backed by Berkeley DB Java Edition. This includes the following:

- Add, delete, modify, and modify DN operations that make changes to JSON objects stored in the server.
- LDIF imports that include JSON objects, including updates to the cache size estimates for the JSON indexes.
- The `rebuild-index` tool and corresponding backend code to make it possible to generate and rebuild indexes for JSON data. It must be possible to build all JSON indexes for all or a specified subset of fields associated with a given attribute type. The `verify-index` tool should also work with JSON indexes to make it possible to check their validity.
- Matching entry count control and debugsearchindex return attribute provide information about relevant JSON index usage.
- Support for monitoring index content and usage.

ⓘ **Note:** Exploded indexes and the entry balancing global index do not support JSON objects.

Working with local DB VLV indexes

Local DB VLV indexes allow a client to request a subset of results from a sorted list that match a specific search base, scope, and filter. The client can navigate through the list by passing a context back to the server with the virtual list view control. The Local DB VLV index can be used only when the client request contains the virtual list view (VLV) control and the client has been authorized with an ACI with a **`targetcontrol`** of **2.16.840.1.113730.3.4.9**.

ⓘ **Note:** A client request, which includes a virtual list view control, can be successfully processed without a matching Local DB VLV index if the search is completely indexed. This is not an efficient means of using VLV, since the server has to retrieve each entry twice.

Viewing the list of local DB VLV indexes

Steps

- Use **`dsconfig`** with the `list-local-db-indexes` option to view the default list of indexes. In the example, no VLV indexes are defined.

```
$ bin/dsconfig list-local-db-vlv-indexes --backend-name userRoot
```

Creating a new local DB VLV index

Steps

1. Use **dsconfig** with the `create-local-db-vlv-index` option and the `--index-name`, `--backend-name`, and `--set index-type:(propertyValue)` options. If you do not set any property values, the default values are assigned.

```
$ bin/dsconfig create-local-db-vlv-index \
  --index-name givenName --backend-name userRoot --set base-
dn:dc=example,dc=com \
  --set scope:whole-subtree --set filter:"(objectclass=*)" \
  --set sort-order:givenName
```

2. Rebuild the index using the **rebuild-index** tool. You must add the “vlv.” prefix to the index name to rebuild the VLV index. The following command can be run with the server on or offline with the addition of the `--task` and connection options.

```
$ bin/rebuild-index --baseDN dc=example,dc=com --index vlv.givenName
```

Modifying a VLV index’s configuration

Steps

1. Use **dsconfig** with the `set-local-db-vlv-index-prop` option and the `--index-name` and `--backend-name` properties. In this example, update the `base-dn` property.

```
$ bin/dsconfig set-local-db-vlv-index-prop --index-name givenName \
  --backend-name userRoot --set base-dn:ou=People,dc=example,dc=com
```

2. Rebuild the index using the **rebuild-index** tool. You must add the prefix "vlv." to the index name. The following command can be run with the server on or offline with the addition of the `--task` and connection options.

```
$ bin/rebuild-index --baseDN dc=example,dc=com --index vlv.givenName
```

Deleting a VLV index

About this task

You can delete a VLV index using the **dsconfig** tool. Check that the index is not being used in any plugin applications before deleting it.

Steps

1. Use **dsconfig** with the `delete-local-db-vlv-index` option to remove it from the database.

```
$ bin/dsconfig delete-local-db-vlv-index --index-name givenName \
  --backend-name userRoot
```

2. Verify the deletion by trying to view the vlv index.

```
$ bin/dsconfig get-local-db-vlv-index-prop --index-name givenName \
  --backend-name userRoot
```

Working with filtered indexes

The PingDirectory Server filtered index is useful when client search requests consisting of a compound &-filter with individual components matching a large number of entries, potentially greater than the index entry limit, have an intersection of a relatively small number of entries.

For example, assume a database contains several thousand company profiles and each company profile is represented by many entries. The "**(objectClass=company)**" filter matches a small set of entries per company, and therefore may exceed the index entry limit since there are many companies. Also assume that the "**(companyDomain=example.com)**" filter matches many of the entries for the company with domain **example.com** and can also result in an unindexed search. The more narrow filter "**(&(objectClass=company)(companyDomain=example.com))**" also results in an unindexed search but only matches a small number of entries. The filtered index makes it possible to index this compound filter by defining an equality index on the **companyDomain** attribute with a static filter of "**(objectClass=company)**" in the **equality-index-filter** property of the index.

Filtered indexing is primarily useful for cases in which clients frequently issue searches with AND filters that meet the following criteria:

- The AND filter itself matches a relatively small number of entries, but each of the individual components may match a very large number of entries.
- The filter has a dynamic component that does change, and that dynamic component always uses the same attribute.
- The filter has a static component that doesn't change.
- The filter must be narrowed to a base DN, for data structures with many branches, or if indexed attribute values appear in a very large number of entries.

The filtered index can be maintained independently from the equality filter for that attribute. Further, the filtered index will be used only for searches containing the equality component with the associated attribute type ANDed with this filter. When configuring a filtered index, be aware of the **equality-index-filter** and **maintain-equality-index-without-filter** properties of the index.

Once configured and built with the **rebuild-index** tool or **import-ldif**, searches with filters based on the above example will be processed with the index:

```
(&(objectClass=company)(companyDomain=example.com))
(&(objectClass=company)(|(companyDomain=example.com)
(companyDomain=example.org)))
(&(companyDomain=example.com)(objectClass=company))
(&(companyDomain=example.com)(&(objectClass=company)))
(&(companyDomain=example.com)(objectClass=company)(something=else))
(&(companyDomain=example.com)(&(objectClass=company)(something=else)))
(|(&(objectClass=company)(companyDomain=example.com))(&(objectClass=company)
(companyDomain=example.org)))
```

When configuring a filtered index, define the following properties:

Filtered Index Properties

Filtered Index Properties	Description
equality-index-filter	Specifies a search filter that may be used in conjunction with an equality component for the associated attribute type. If an equality index filter is defined, then an additional equality index will be maintained for the associated attribute, but only for entries that match the provided filter. Further, the index will be used only for searches containing an equality component with the associated attribute type ANDed with this filter.
maintain-equality-index-without-filter	Specifies whether to maintain a separate equality index for the associated attribute without any filter, in addition to maintaining an index for each equality index filter that is defined. If this is false, then the attribute will not be indexed for equality by itself but only in conjunction with the defined equality index filters.

Creating a filtered index

Steps

1. Use the **dsconfig** tool to create a filtered index. The following command creates an equality index on the `companyDomain` attribute and maintains an index for the equality filter defined "`(objectclass=company)`". After you have created the index, you must rebuild the indexes.

```
$ bin/dsconfig create-local-db-index --backend-name "userRoot" \
  --index-name companyDomain --set maintain-equality-index-without-
filter:true \
  --set index-type:equality --set equality-index-
filter:"(objectclass=company)"
```

2. Stop the Directory Server using **bin/stop-server**.
3. Run the **rebuild-index** tool.

```
$ bin/rebuild-index --baseDN dc=example,dc=com --index companyDomain
```

4. Start the Directory Server using **bin/start-server**.

Tuning indexes

The PingDirectory Server provides several tools to help you optimize your indexes and improve the overall read and write performance for your system. The server now supports an optional expanded index database using an *exploded* format to process high write load operations. To view the current state of the server's indexes and make adjustments to the index databases, the Directory Server automatically generates an Index Summary Statistics Table after each LDIF import or index rebuild. The **dbtest** tool also includes an Index Histogram to determine the key datasize for the indexes. This section provides descriptions of each of these tools.

About the exploded index format

The `index-entry-limit` Backend configuration property specifies the maximum number of entries kept in an index record, before the server stops maintaining that record, begins scanning the whole database, and runs an expensive unindexed search. If any index keys have already reached this limit, indexes must be rebuilt before they can be allowed to use a new limit. If `index-entry-limit` is configured to be larger than 50000, then any keys that match more than 50000 entries will be stored in a separate database in an expanded (or "exploded") format.

All keys whose entry count is less than 50000 continue to be stored in one database in a consolidated format, such that changes to the key require rewriting all the entry IDs matching the key. All keys whose entry count is greater than 50000 and less than the `index-entry-limit` are stored in a separate database in an exploded format, such that changes to the key require writing only to the updated entry ID.

If the `index-entry-limit` is increased to 100000, any key with an entry count less than 50000 continues to be stored in consolidated format. If a key has an entry count greater than 50000, it will be stored in a separate database where each key is stored with its entry ID individually. The consolidated format is very efficient for read operations because the server can retrieve a row of entry IDs at once, while the exploded format is far more efficient for high volumes of write operations since it avoids large on-disk growth.

Monitoring index entry limits

Index keys that have reached their limit require indexes to be rebuilt before they can be allowed to use a new limit. There are several ways to monitor index limits to avoid a potentially costly rebuild. There are cases in which it is acceptable for index keys to exceed the index entry limit. For example, the `objectClass` attribute type should be indexed for equality because the server needs to use it to find all group entries when bringing a backend online, and also because applications frequently need to find entries of a specific type. However, it doesn't make sense for the "top" objectClass key to be indexed, because it appears in every entry in the server.

Choose an index entry limit value that is high enough to ensure that all of the right keys are indexed, but keys that occur too frequently are not. The `verify-index` tool `--listKeysNearestIndexEntryLimt` argument lists a specified number of keys that are closest to the limit without having exceeded it. The index entry limit should be larger than the number of entries matching the largest key to remain indexed, with enough overhead to account for future growth. Use this command regularly to determine if the index entry limit needs to be adjusted.

The `verify-index` tool also provides the `--listKeysExceedingIndexEntryLimit` argument to list all keys for which the value has exceeded the index entry limit and the number of entries in which they appear. If there are keys for which the limit has already exceeded but that need to be maintained, adjust the index entry limit to be higher than the number of entries that contain that key (with additional room for future growth) and run the `rebuild-index` tool (or export to LDIF and re-import).

The server provides other ways to determine if index keys have exceeded, or are close to exceeding, the index entry limit. Some of these include:

- When performing an LDIF import, the tool includes an "Index Summary Statistics" section that provides usage information for each index, including the number of keys for which the index entry limit has been exceeded, and also the number of keys for which the number of matching entries falls within a number of predefined buckets (such as 1–4 entries, 5–9 entries, 10–99 entries, and 100–999 entries).
- If, during a search operation, the server accesses one or more index keys whose values have exceeded the index entry limit, the access log message for that operation will include an `indexesWithKeysAccessedExceedingEntryLimit` field containing a comma-delimited list of the appropriate indexes. The same access log field may appear in log messages for add, delete, modify, and modify DN operations in which the server wrote to, or tried to write to, at least one index key whose value exceeded the index entry limit.
- If, during a search operation, the server accesses one or more index keys whose values have not yet exceeded the index entry limit but are more than 80 percent to reaching that limit, the access log message for that operation will include an `indexesWithKeysAccessedNearEntryLimit` field containing a comma-delimited list of the appropriate indexes. The same access log field may appear in log messages for add, delete, modify, and modify DN operations in which the server wrote to at least one index key whose value was within 80 percent of the index entry limit.
- If a search operation requests either includes the `debugsearchindex` attribute, or the matching entry count request control with debugging enabled, the debug information will include any indexes accessed that have exceeded the index entry limit, or that are within 80 percent of the configured index entry limit.
- The monitor entry for each configured index includes attributes that provide information about the number of index keys that have been encountered (since the backend was brought online, or since the index entry limit was changed) in a number of different categories. These monitor attributes include:
  - `ds-index-exceeded-entry-limit-count-since-db-open` — The number of index keys for which the number of matching entries has crossed the index entry limit due to a write operation.
  - `ds-index-unique-keys-near-entry-limit-accessed-by-search-since-db-open` — The number of unique index keys that have been accessed by a search operation for which the number of matching entries is within 80 percent of the index entry limit.
  - `ds-index-unique-keys-exceeding-entry-limit-accessed-by-search-since-db-open` — The number of unique index keys that have been accessed by a search operation for which the number of matching entries has exceeded the index entry limit at some point since the index was last built.
  - `ds-index-unique-keys-near-entry-limit-accessed-by-write-since-db-open` — The number of unique index keys that have been accessed by a write operation for which the number of matching entries is within 80 percent of the index entry limit.
  - `ds-index-unique-keys-exceeding-entry-limit-accessed-by-write-since-db-open` — The number of unique index keys that have been accessed by a write operation for which the number of matching entries has exceeded the index entry limit at some point since the index was last built.

About the dbtest Index Status table

The **`dbtest`** tool has a **`list-all --analyze`** option that generates the current status of all of the databases on your system, including all index databases. The table shows the type, entry count (i.e., the number of records in the dtabase), index status (TRUSTED to indicate that the indexes are up-to-date, or UNTRUSTED if the index needs rebuilding), the total data size for each key, the average data size for each key and the maximum data size for each key. Note also that any indexes that are in exploded format are listed on this table.

Index Name	Index Type	JE Database Name	Index Status
id2children	Index	dc_example_dc_com_id2children	TRUSTED
id2subtree	Index	dc_example_dc_com_id2subtree	TRUSTED
uid.equality	Index	dc_example_dc_com_uid.equality	TRUSTED
aci.presence	Index	dc_example_dc_com_aci.presence	TRUSTED
ds-soft-delete-timestamp.ordering	Index	dc_example_dc_com_ds-soft-delete-timestamp.ordering	TRUSTED
ds-soft-delete-from-dn.equality	Index	dc_example_dc_com_ds-soft-delete-from-dn.equality	TRUSTED
givenName.equality	Index	dc_example_dc_com_givenName.equality	TRUSTED
givenName.substring	Index	dc_example_dc_com_givenName.substring	TRUSTED
objectClass.equality	Index	dc_example_dc_com_objectClass.equality	TRUSTED
member.equality	Index	dc_example_dc_com_member.equality	TRUSTED
uniqueMember.equality	Index	dc_example_dc_com_uniqueMember.equality	TRUSTED
cn.equality	Index	dc_example_dc_com_cn.equality	TRUSTED
cn.substring	Index	dc_example_dc_com_cn.substring	TRUSTED
sn.equality	Index	dc_example_dc_com_sn.equality	TRUSTED
sn.substring	Index	dc_example_dc_com_sn.substring	TRUSTED
telephoneNumber.equality	Index	dc_example_dc_com_telephoneNumber.equality	TRUSTED
mail.equality	Index	dc_example_dc_com_mail.equality	TRUSTED
ds-entry-unique-id.equality	Index	dc_example_dc_com_ds-entry-unique-id.equality	TRUSTED

MY TITLE dbtest Output Including Index Databases

Configuring the index properties

About this task

By default, the `index-entry-limit` is set to 4000, which means the server will stop maintaining index values for keys that match more than 4000 entries. This can be changed with the **`dsconfig`** tool.

Before running the following commands, be aware that you will need to do an index rebuild on the system, which requires a system shutdown, unless the command is run as a task.

To configure the index properties:

Steps

1. Run **`dsconfig`** and set the `index-entry-limit` to 5000. By default, the value is set to 4000. Remember to include the bind parameters for your system. Once you enter the command, confirm that you want to apply the changes.

```
$ bin/dsconfig set-backend-prop \
  --backend-name userRoot \
  --set index-entry-limit:5000 \
One or more configuration property changes require administrative action
or confirmation/notification. Those properties include:
* index-entry-limit: If any index keys have already reached this limit,
indexes must be rebuilt before they will be allowed to use the new limit.
Setting a large limit (greater than 10,000) could have a big impact on
write performance and database growth on disk.
Continue? Choose 'no' to return to the previous step (yes / no) [yes]: yes
```

2. Stop the server.

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

3. Rebuild the index.

```
$ bin/rebuild-index --baseDN dc=example,dc=com \
  --index cn --index givenName --index objectClass \
  --index sn --maxThreads 10
```

4. View the Index Summary Statistics table, which is automatically displayed to system out after running the **`rebuild-index`** command. You can also access the table at `logs/tools/rebuild-index-summary.txt`. Repeat the last three steps to make more adjustments to your indexes.

```
          --- Index Summary Statistics ---
Index                                 : Limit : >Limit : Max    : 1-9    : 10-99 : 100-999 : 1000-9999 : 10000-99999 : 100000-999999
--------------------------------------:-------:--------:--------:--------:-------:---------:-----------:-------------:--------------
dc_example_dc_com_cn.equality         : 5000  :        : 1      : 100001 :       :         :           :             :
dc_example_dc_com_cn.substring        : 5000  : 9      : 15401  : 131746 : 22372 : 545     : 39        : 3           :
dc_example_dc_com_givenName.equality  : 5000  :        : 16     : 3788   : 4817  :         :           :             :
dc_example_dc_com_givenName.substring : 5000  : 8      : 23809  : 7039   : 12062 : 483     : 42        : 3           :
dc_example_dc_com_objectClass.equality: 5000  : 4      : 100003 : 3      :       :         :           :             : 4
dc_example_dc_com_sn.equality         : 5000  :        : 8      : 13419  :       :         :           :             :
dc_example_dc_com_sn.substring        : 5000  : 9      : 15401  : 33967  : 7055  : 459     : 39        : 3           :

The first three columns of numbers provide (1) "Limit" - the index entry limit (or blank if there is no limit),
(2) ">Limit" - the number of keys whose entry count exceeds the entry limit, and (3) "Max" - the
maximum entry count for any key in the index. The remaining columns provide the number of keys
whose entry count falls in the range indicated in the column heading
```

5. Restart the server.

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

About the Index Summary Statistics table

The Directory Server now automatically generates an Index Summary Statistics Table, which can be used to determine the optimal configuration for your system's indexes. This table is only generated when the `rebuild-index` tool is run in offline mode. The table is generated after any LDIF import or an index rebuild, and is written to system out and to `logs/tools/rebuild-index-summary.txt`. The table lists the current index entry limit (set by the `index-entry-limit` property on the local DB configuration), the number of keys whose entry count exceeds this limit if any, and the maximum entry count for any key in the index. The table then displays a histogram of the number of keys whose entry falls within a range of values. An example of the Index Summary Statistics table is show below for the `sn.equality` and the `sn.substring` indexes.

The following figure shows that there are seven substrings whose entry counts exceed the `index-entry-limit` of 4000. Six of the substrings are in the 10000-99999 range with the maximum entry count being 13419. By deduction, one more substring must be present in the 1000-9999 range that exceeds the `index-entry-limit` of 4000. These substrings could be expensive for search operations.

```
          --- Index Summary Statistics ---
Index                            : Limit : >Limit : Max   : 1-9    : 10-99 : 100-999 : 1000-9999 : 10000-99999
---------------------------------:-------:--------:-------:--------:-------:---------:-----------:------------
dc_example_dc_com_sn.equality    : 4000  :        : 2     : 100000 :       :         :           :
dc_example_dc_com_sn.substring   : 4000  : 7      : 13419 : 150814 : 7109  : 407     : 36        : 6
```

MY TITLE Example of an Index Summary Statistics Table

Managing Entries

The Directory Server is a fully LDAPv3-compliant server that comes with a comprehensive set of LDAP command-line tools to search, add, modify, and delete entries.

This chapter presents the following topics:

Searching entries

The Directory Server provides an **`ldapsearch`** tool to search for entries or attributes within your server. The tool requires the LDAP connection parameters needed to bind to the server, including the `baseDN` option to specify the starting point of the search within the server, and the search scope. The `searchScope` option determines the depth of the search:

- `base` (search only the entry specified)

- `one` (search only the children of the entry and not the entry itself)
- `sub` (search the entry and its descendents)

The **`ldapsearch`** tool provides basic functionality as specified by the RFC 2254 but provides additional features that takes advantage of the Directory Server's control mechanisms. For more information, run the **`ldapsearch`** `--help` function.

Searching the root DSE

About this task
The Root DSE is a special entry that resides at the root of the directory information tree (DIT). The entry holds operational information about the server and its supported controls. Specifically, the root DSE entry provides information about the supported LDAPv3 controls, SASL mechanisms, password authentication schemes, supported LDAP protocols, additional features, naming contexts, extended operations, and server information.

ⓘ **Note:** The Directory Server provides an option to retrieve the Root DSE's operational attributes and add them to the user attribute map of the generated entry. This feature allows client applications that have difficulty handling operational attributes to access the root DSE using the `show-all-attributes` configuration property. Once this property is set, the associated attribute types are re-created and re-registered as user attributes in the schema (in memory, not on disk). Once you set the property, you can use **`ldapsearch`** without "+" to view the root DSE.

Use the **`dsconfig`** tool to set the `show-all-attributes` property to TRUE, as follows:

```
$ bin/dsconfig set-root-dse-backend-prop --set show-all-attributes:true
```

Steps

- Use **`ldapsearch`** to view the root DSE entry on the Directory Server. Be sure you include the "+" to display the operational attributes in the entry.

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

Searching all entries in the Directory Server

Steps

- Use **`ldapsearch`** to search all entries in the Directory Server. The filter `"(objectclass=*)"` matches all entries. If the `--searchScope` option is not specified, the command defaults to a search scope of sub:

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

Searching for an access control instruction

Steps

- Use **`ldapsearch`** to search the `dc=example,dc=com` base DN entry. The filter `"(aci=*)"` matches all `aci` attributes under the base DN, and the `aci` attribute is specified so that only it is returned. The `cn=Directory Manager` bind DN has the privileges to view an ACI.

  ```
  $ bin/ldapsearch --baseDN dc=example,dc=com "(aci=*)" aci
  ```

  ```
  dn: dc=example,dc=com
  aci: (targetattr!="userPassword")
    (version 3.0; acl "Allow anonymous read access for anyone";
  ```

```
    allow (read,search,compare) userdn="ldap:///anyone";)
aci: (targetattr="*")
  (version 3.0; acl "Allow users to update their own entries";
    allow (write) userdn="ldap:///self";)
aci: (targetattr="*")
  (version 3.0; acl "Grant full access for the admin user";
    allow (all) userdn="ldap:///uid=admin,dc=example,dc=com";)
```

Searching for the schema

Steps

- Use **ldapsearch** to search the `cn=schema` entry. The base DN is specified as `cn=schema`, and the filter "`(objectclass=*)`" matches all entries. The command uses a special attribute "+" to return all operational attributes:

```
$ bin/ldapsearch --baseDN cn=schema \
  --searchScope base "(objectclass=*)" "+"
```

Searching for a single entry using base scope and base DN

Steps

- Use **ldapsearch** to search for a single entry by specifying the base scope and DN:

```
$ bin/ldapsearch --baseDN uid=user.14,ou=People,dc=example,dc=com \
  --searchScope base "(objectclass=*)"
```

Searching for a single entry using the search filter

Steps

- Search for a single entry by specifying the `sub` scope and a search filter that describes a single entry:

```
$ bin/ldapsearch --baseDN ou=People,dc=example,dc=com \
  --searchScope sub "(uid=user.14)"
```

Searching for all immediate children for restricted return values

Steps

- Search for all immediate children of `ou=People,dc=example,dc=com`. The attributes returned are restricted to `sn` and `givenName`. The special attribute "+" returns all operational attributes:

```
$ bin/ldapsearch --baseDN ou=People,dc=example,dc=com \
  --searchScope one '(objectclass=*)' sn givenName "+"
```

Searching for all children of an entry in sorted order

Steps

- Search for all children of the `ou=People,dc=example,dc=com` subtree. The resulting entries are sorted by the server in ascending order by `sn` and then in descending order by `givenName`:

```
$ bin/ldapsearch --baseDN ou=People,dc=example,dc=com \
  --searchScope sub --sortOrder sn,-givenName '(objectclass=*)'
```

Limiting the number of returned search entries and search time

Steps

- Search for a subset of the entries in the `ou=People,dc=example,dc=com` subtree by specifying a compound filter. No more than 200 entries will be returned and the server will spend no more than 5 seconds processing the request. Returned attributes are restricted to a few operational attributes:

```
$ bin/ldapsearch --baseDN ou=People,dc=example,dc=com \
  --searchScope sub --sizeLimit 200 --timeLimit 5 \
  "(&(sn<=Doe)(employeeNumber<=1000))" ds-entry-unique-id entryUUID
```

Getting information about how indexes are used in a search operation

The PingDirectory Server uses indexes to improve database search performance and provide consistent search rates regardless of the number of database objects stored in the Directory Information Tree (DIT). Information about the can be obtained about the server's use of indexes in the course of processing a search operation. The following are two basic ways to accomplish this.

Issue a search request with the desired base DN, scope, and filter, and request that the server return the special **`debugsearchindex`** attribute. Users will need to be granted access to the **`debugsearchindex`** operational attribute and the **`cn=debugsearch`** portion of the DIT with the following command:

```
$ bin/dsconfig set-access-control-handler-prop \
 --add "global-aci:(targetattr=\"debugsearchindex\")(target=\"ldap:///
cn=debugsearch\")
    (version 3.0; acl \"Allow members of the Index Debugging Users group
    to request the debugsearchindex operational attribute \"; allow
    (read,search,compare) groupdn=\"ldap:///cn=Index Debugging
    Users,ou=Groups,dc=example,dc=com\";)"
```

To issue a search request for the server return the special **`debugsearchindex`** attribute, use the **`ldapsearch`** command such as:

```
$ bin/ldapsearch --hostname ds.example.com \
 --port 389 --bindDN uid=admin,dc=example,dc=com \
 --bindPassword password \
 --baseDN dc=example,dc=com \
 --searchScope sub "(&(givenName=John)(sn=Doe))" debugsearchindex
 dn: cn=debugsearch
 debugsearchindex: 0.040 ms - Beginning index processing for search
     request with base DN 'dc=example,dc=com', scope wholeSubtree,
     and filter (&(givenName=John)(sn=Doe)).
 debugsearchindex: 0.067 ms - Unable to optimize the AND filter
     beyond what the client already provided.
 debugsearchindex: 0.834 ms - Candidate set obtained for single-key
     filter (givenName=John) from index dc_example_dc_com_givenName.equality.
     Candidate set:  CandidateSet(isDefined=true, isExploded=false,
     isResolved=true, size=2, originalFilter=(givenName=John),
     remainingFilter=null, matchingEntryCountType=UNEXAMINED_COUNT)
 debugsearchindex: 0.030 ms - Final candidate set for filter (givenName=John)
     obtained from an unexploded index key in
 dc_example_dc_com_givenName.equality.
     Since the scope of the search includes the entire entry container, there
 is
     no need to attempt to further pare down the results based on the search
 scope.
     Candidate set:  CandidateSet(isDefined=true, isExploded=false,
 isResolved=true,
     size=2, originalFilter=(givenName=John), remainingFilter=null,
     matchingEntryCountType=UNEXAMINED_COUNT)
 debugsearchindex: 0.020 ms - Short-circuiting index processing for AND filter
```

```
    (&(givenName=John)(sn=Doe)) after evaluating single-key component
    (givenName=John) because the current ID set size of 2 is within the
     short-circuit threshold of 5.
debugsearchindex: 0.030 ms - Obtained a candidate set of size 2 for AND
filter
    (&(givenName=John)(sn=Doe)) with remaining filter (sn=Doe).  Even though
    there is still more of the filter to evaluate, the current candidate set
is
    within the short-circuit threshold of 5, so no additional index
processing will
    be performed to try to pare down the results based on the remaining
filter or the
    search scope.  Candidate set:  CandidateSet(isDefined=true,
isExploded=false,
    isResolved=true, size=2, originalFilter=(&(givenName=John)(sn=Doe)),
    remainingFilter=(sn=Doe), matchingEntryCountType=UPPER_BOUND)
debugsearchindex: 0.016 ms - Completed all index processing.  Candidate set:
    CandidateSet(isDefined=true, isExploded=false, isResolved=true, size=2,
    originalFilter=(&(givenName=John)(sn=Doe)), remainingFilter=(sn=Doe),
    matchingEntryCountType=UPPER_BOUND)
```

The second way would be to issue a search request with the desired base DN, scope, and filter, and include the matching entry count request control with the debug option set to **`true`**. To do this with **`ldapsearch`**, use a command such as:

```
$ bin/ldapsearch --hostname ds.example.com --port 389 \
 --bindDN uid=admin,dc=example,dc=com --bindPassword password \
 --baseDN dc=example,dc=com \
 --searchScope sub --matchingEntryCountControl examineCount=0:debug
 "(&(givenName=John)(sn=Doe))"
 Upper Bound on Matching Entry Count:  2
 Matching Entry Count Debug Messages:
 * naw-desktop:1389 - 0.104 ms - Beginning index processing for search request
 with
    base DN 'dc=example,dc=com', scope wholeSubtree, and filter
 (&(givenName=John)(sn=Doe)).
 * naw-desktop:1389 - 0.105 ms - Unable to optimize the AND filter beyond what
 the client already provided.
 * naw-desktop:1389 - 0.614 ms - Candidate set obtained for single-key filter
 (givenName=John) from index
    dc_example_dc_com_givenName.equality.  Candidate set:
    CandidateSet(isDefined=true, isExploded=false, isResolved=true,
    size=2, originalFilter=(givenName=John), remainingFilter=null,
 matchingEntryCountType=UNEXAMINED_COUNT)
 * naw-desktop:1389 - 0.090 ms - Final candidate set for filter
    (givenName=John) obtained from an unexploded index key in
    dc_example_dc_com_givenName.equality.  Since the scope of the search
    includes the entire entry container, there is no need
    to attempt to further pare down the results based on the search scope.
    Candidate set:  CandidateSet(isDefined=true, isExploded=false,
 isResolved=true,
    size=2, originalFilter=(givenName=John), remainingFilter=null,
 matchingEntryCountType=UNEXAMINED_COUNT)
 * naw-desktop:1389 - 0.045 ms - Short-circuiting index processing for AND
 filter
    (&(givenName=John)(sn=Doe)) after evaluating single-key component
    (givenName=John) because the current ID set size of 2 is within the short-
circuit threshold of 5.
 * naw-desktop:1389 - 0.111 ms - Obtained a candidate set of size 2 for AND
 filter
    (&(givenName=John)(sn=Doe)) with remaining filter (sn=Doe).  Even though
 there is
    still more of the filter to evaluate, the current candidate set is within
```

```
   the short-circuit threshold of 5, so no additional index processing will
be performed
   to try to pare down the results based on the remaining filter or the
search scope.
   Candidate set:  CandidateSet(isDefined=true, isExploded=false,
isResolved=true, size=2,
   originalFilter=(&(givenName=John)(sn=Doe)), remainingFilter=(sn=Doe),
   matchingEntryCountType=UPPER_BOUND)
* naw-desktop:1389 - 0.040 ms - Completed all index processing.  Candidate
set:
   CandidateSet(isDefined=true, isExploded=false, isResolved=true, size=2,
   originalFilter=(&(givenName=John)(sn=Doe)), remainingFilter=(sn=Doe),
   matchingEntryCountType=UPPER_BOUND)
* naw-desktop:1389 - The search is partially indexed
(candidatesAreInScope=true,
   unindexedFilterPortion=(sn=Doe))
* naw-desktop:1389 - Constructing an UPPER_BOUND response with a count of 2
```

Working with the matching entry count control

The **ldapsearch** command can be used with the **--matchingEntryCountControl** option to determine the count of entries that match a search filter. Users will need to be granted access to this control by its OID **1.3.6.1.4.1.30221.2.5.36** with the following command:

```
$ bin/dsconfig set-access-control-handler-prop \
 --add "global-aci:(targetcontrol=\"1.3.6.1.4.1.30221.2.5.36\")
   (version 3.0; acl \"Allow members of the Index Debugging Users group to
   use the matching entry count request control \"; allow (read)
   groupdn=\"ldap:///cn=Index Debugging Users,ou=Groups,dc=example,dc=com\";)"
```

An **examineCount** control can be used for searches that are at least partially indexed, in order to determine whether to return an examined count, an unexamined count, or an upper bound count. The factors that determine what is returned are:

- A search is fully indexed if indexes can be used to identify the entry IDs for all entries that match the filter without ambiguity. Indexes can also be used to make sure that all of those candidates are within the scope of the search.
- A search is partially indexed if indexes can be used to identify the entry IDs for all entries that match the search criteria, but the candidate list may potentially also include entries that either don't match the filter or are outside the scope of the search.
- A search is unindexed if it is not possible to retrieve a candidate list based on either the filter or the search scope.
- An unexamined count is a count of the exact number of entries that match the search criteria, only through the use of index processing.
- An examined count is the same as an unexamined count, except that all of the candidate entries are examined to determine whether they would have been returned to the client. An examined count may be less than an unexamined count if the set of matching entries includes those that would be removed by access control evaluation, or special entries like LDAP sub-entries, replication conflict entries, or soft-deleted entries.
- An upper bound count is the maximum number of entries that match the criteria, but indicates that the server could not determine exactly how many matching entries there were without examining each candidate, which it did not do.
- If a search is fully indexed, the result is an examined count or an unexamined count. If **alwaysExamine** is true and **examineCount** is greater than or equal to the number of candidates, the result is an examined count. If **alwaysExamine** is false, or if the number of candidates exceeds **examineCount**, the result is an unexamined count.
- If a search is partially indexed, the result is either an examined count or an upper bound count. The **alwaysExamine** flag isn't relevant in this case. If **examineCount** is greater than or equal to the number of candidates, the result is an examined count. If not, the result is an upper bound count.

- If a search is unindexed, the result is either an examined count or an unknown count. If **allowUnindexed** is true, the unindexed search is processed, which can be very expensive. Instead of getting the matching entries back, the examined count is returned. If **allowUnindexed** is false, an unknown count is returned. If **allowUnindexed** is true, the requester needs to have the **unindexed-search** privilege to get the exact count.

The following is a sample `ldapsearch` with the `--matchingEntryCountControl` option:

```
$./ldapsearch \
  --bindDN "cn=directory manager" \
  --bindPassword password \
  --baseDN dc=example,dc=com \
  --matchingEntryCountControl
 examineCount=100:alwaysExamine:allowUnindexed:debug \
  "(objectclass=*)"
```

Adding entries

Depending on the number of entries that you want to add to your Directory Server, you can use the **ldapmodify** tool for small additions. The **ldapmodify** tool provides two methods for adding a single entry: using a LDIF file or from the command line. The attributes must conform to your schema and contain the required object classes.

Adding requests with the **ignore-no-user-modification** control enable a client to include attributes that are not normally allowed from external sources. For example, the **userPassword** attribute is a user-modifiable attribute. An add request with the **ignore-no-user-modification** control allows a one-time exception to the password policy, even if the requesting client does not have the **bypass-pw-policy** privilege. This exception enables specifying pre-encoded passwords.

ⓘ **Note:** When adding an entry, the server can ensure that the entry's RDN is unique and does not contain any sensitive information by replacing the provided entry's RDN with the server-generated entryUUID value. An LDAP client written with the LDAP SDK for Java can use the **NameWithEntryUUIDRequestControl** to explicitly indicate which add requests should be named in this way, or the **ldapmodify** tool with the **--nameWithEntryUUID** argument. Also, the **auto-name-with-entry-uuid-connection-criteria** and **auto-name-with-entry-uuid-request-criteria** global configuration properties can be used to identify which add requests should be automatically named this way.

The uniqueness request control can also be used with **ldapmodify** for enforcing uniqueness on a per-request basis. Provide at least one of the **uniquenessAttribute** or **uniquenessFilter** arguments with the request. For more information about this control, see the LDAP SDK documentation and the **com.unboundid.ldap.sdk.unboundidds.controls.UniquenessResponseControl** class for using the control.

Adding an entry using an LDIF file

Steps

1. Open a text editor and create an entry that conforms with your schema. For example, add the following entry in the file and save the file as `add-user.ldif`. For the `userPassword` attribute, enter the cleartext password. The Directory Server encrypts the password and stores its encrypted value in the server. Make sure that the LDIF file has limited read permissions for only authorized administrators.

```
dn: uid=user.2000,ou=People,dc=example,dc=com
objectClass: top
objectClass: person
objectClass: organizationalPerson
objectClass: inetOrgPerson
```

```
postalAddress: Toby Hall$73600 Mash Street$Cincinnati, OH 50563 postalCode:
 50563
description: This is the description for Toby Hall.
uid: user.2000
userPassword: wordsmith employeeNumber: 2000
initials: TBH
givenName: Toby
pager: +1 596 232 3321
mobile: +1 039 311 9878
cn: Toby Hall
sn: Hall
telephoneNumber: +1 097 678 9688
street: 73600 Mash Street
homePhone: +1 214 233 8484
l: Cincinnati
mail: user.2000@maildomain.net
st: OH
```

2. Use the **`ldapmodify`** tool to add the entry specified in the LDIF file. You will see a confirmation message of the addition. If the command is successful, you will see generated success messages with the "#" symbol.

```
$ bin/ldapmodify --defaultAdd --filename add-user.ldif
```

```
# Processing ADD request for uid=user.2000,ou=People,dc=example,dc=com
# ADD operation successful for DN uid=user.2000,ou=People,dc=example,dc=com
```

Adding an entry using the changetype LDIF directive

About this task

RFC 2849 specifies LDIF directives that can be used within your LDIF files. The most commonly used directive is `changetype`, which follows the `dn:` directive and defines the operation on the entry. The main advantage of using this method in an LDIF file is that you can combine adds and modifies in one file.

Steps

1. Open a text editor and create an entry that conforms with your schema. For example, add the following entry in the file and save the file as `add-user2.ldif`. Note the use of the `changetype` directive in the second line.

```
dn: uid=user.2001,ou=People,dc=example,dc=com
changetype: add
objectClass: top
objectClass: person
objectClass: organizationalPerson
objectClass: inetOrgPerson
postalAddress: Seely Dorm$100 Apple Street$Cincinnati, OH 50563
postalCode: 50563
description: This is the description for Seely Dorm.
uid: user.2001
userPassword: pleasantry
employeeNumber: 2001
initials: SPD
givenName: Seely pager: +1 596 665 3344
mobile: +1 039 686 4949
cn: Seely Dorm
sn: Dorm
telephoneNumber: +1 097 257 7542
street: 100 Apple Street
homePhone: +1 214 521 4883
l: Cincinnati
```

```
mail: user.2001@maildomain.net
st: OH
```

2. Use the **ldapmodify** tool to add the entry specified in the LDIF file. You will see a confirmation message of the addition. In this example, you do not need to use the `--defaultAdd` or its shortform `-a` option with the command.

```
$ bin/ldapmodify --filename add-user2.ldif
```

Adding multiple entries in a single file

About this task

You can have multiple entries in your LDIF file by simply separating each DN and its entry with a blank line from the next entry.

Steps

1. Open a text editor and create a couple of entries that conform to your schema. For example, add the following entries in the file and save the file as `add-user3.ldif`. Separate each entry with a blank line.

```
dn: uid=user.2003,ou=People,dc=example,dc=com
objectClass: top
objectClass: person
objectClass:
organizationalPerson
objectClass: inetOrgPerson
...(similar attributes to previous examples)...

dn: uid=user.2004,ou=People,dc=example,dc=com
objectClass: top
objectClass: person
objectClass: organizationalPerson
objectClass: inetOrgPerson
...(similar attributes to previous examples)...
```

2. Use the **ldapmodify** tool to add the entries specified in the LDIF file. In this example, we use the short form arguments for the **ldapmodify** tool.

   The `-h` option specifies the host name, the `-p` option specifies the LDAP listener port, `-D` specifies the bind DN, `-w` specifies the bind DN password, `-a` specifies that entries that omit a changetype will be treated as add operations, and `-f` specifies the path to the input file. If the operation is successful, you will see commented messages (those begining with "#") for each addition.

```
$ bin/ldapmodify -h server.example.com -p 389 \
  -D "cn=admin,dc=example,dc=com" -w password -a -f add-user3.ldif
```

```
# Processing ADD request for uid=user.2003,ou=People,dc=example,dc=com
# ADD operation successful for DN uid=user.2003,ou=People,dc=example,dc=com
# Processing ADD request for uid=user.2004,ou=People,dc=example,dc=com
# ADD operation successful for DN uid=user.2004,ou=People,dc=example,dc=com
```

Deleting entries using ldapdelete

You can delete an entry using the **ldapdelete** tool. You should ensure that there are no child entries below the entry as that could create an orphaned entry. Also, make sure that you have properly backed up your system prior to removing any entries.

Deleting an entry using ldapdelete

Steps

Use **ldapdelete** to delete an entry. The following example deletes the `uid=user.14` entry.

```
$ bin/ldapdelete uid=user.14,ou=People,dc=example,dc=com
```

Deleting multiple entries using an LDIF file

Steps

1. You can generate a file of DNs that you would like to delete from the Directory Server.

   The following command searches for all entries in the `ou=Accounting` branch and returns the DNs of the subentries.

   ```
   $ bin/dump-dns -D "cn=admin,dc=example,dc=com" -w password --baseDN \
                          "ou=Accounting,ou=People,dc=example,dc=com" --
   outputFile /usr/local/entry_dns.txt
   ```
2. Run the **ldapdelete** command with the file to delete the entries. The command uses the `--continueError` option, which will continue deleting through the whole list even if an error is encountered for a DN entry.

   ```
   $ bin/ldapdelete --filename /usr/local/entry_dns.txt --continueError
   ```

Deleting entries using ldapmodify

About this task

You can use the LDIF changetype directive to delete an entry from the Directory Server using the **ldapmodify** tool. You can only delete leaf entries.

To delete an entry using **ldapmodify**:

Steps

- From the command line, use the **ldapmodify** tool with the `changetype:delete` directive. Enter the DN, press **Enter** to go to the next line, then enter the changetype directive. Press **Control-D** twice to enter the EOF sequence (UNIX) or **Control-Z** (Windows).

  ```
  $ bin/ldapmodify --hostname server1.example.com -port 389 --bindDN
   "uid=admin,dc=example,dc=com" --bindPassword password
  dn:uid=user.14,ou=People,dc=example,dc=com
  changetype: delete
  ```

Modifying entries using ldapmodify

You can use the **ldapmodify** tool to modify entries from the command line or by using an LDIF file that has the `changetype:modify` directive and value. If you have more than one change, you can separate them using the - (dash) symbol.

Modifying an attribute from the command line

Steps

1. Use the **ldapsearch** tool to locate a specific entry.

   ```
   $ bin/ldapsearch -h server.example.com -p 389 -D
    "cn=admin,dc=example,dc=com" \
   ```

```
  -w password -b dc=example,dc=com "(uid=user.2004)"
```

2. Use the **ldapmodify** command to change attributes from the command line. Specify the modification using the `changetype:modify` directive, and then specify which attributes are to be changed using the `replace` directive. In this example, we change the telephone number of a specific user entry. When you are done typing, you can press **CTRL-D** (Unix EOF escape sequence) twice or **CTRL-Z** (Windows) to process the request.

```
$ bin/ldapmodify -h server.example.com -p 389 -D
 "cn=admin,dc=example,dc=com" \
  -w password
dn: uid=user.2004,ou=People,dc=example,dc=com
changetype: modify
replace: telephoneNumber
telephoneNumber: +1 097 453 8232
```

Modifying multiple attributes in an entry from the command line

Steps

1. Use the **ldapsearch** tool to locate a specific entry.

```
$ bin/ldapsearch -h server.example.com -p 389 -D
 "cn=admin,dc=example,dc=com" \
  -w password -b dc=example,dc=com "(uid=user.2004)"
```

2. Use the **ldapmodify** command to change attributes from the command line. Specify the modification using the `changetype:modify` directive, and then specify which attributes are to be changes using the add and replace directive.

   In this example, we add the `postOfficeBox` attribute, change the mobile and telephone numbers of a specific user entry. The `postOfficeBox` attribute must be present in your schema to allow the addition. The three changes are separated by a dash ("-"). When you are done typing, you can press **CTRL-D** (Unix EOF escape sequence) twice or **CTRL-Z** (Windows) to process the request.

```
$ bin/ldapmodify -h server.example.com -p 389 -D
 "cn=admin,dc=example,dc=com" -w password
dn: uid=user.2004,ou=People,dc=example,dc=com
changetype: modify
add: postOfficeBox
postOfficeBox: 111
-
replace: mobile
mobile: +1 039 831 3737
-
replace: telephoneNumber
telephoneNumber: +1 097 453 8232
```

Adding an attribute from the command line

Steps

- Use the **ldapmodify** command from the command line. Specify the modification using the `changetype:modify` directive, and then specify which attributes are to be added using the `add` directive. In this example, we add another value for the `cn` attribute, which is multi-valued. When you are done typing, you can press **CTRL-D** (UNIX EOF escape sequence) twice to process the request.

```
$ bin/ldapmodify -h server.example.com -p 389 -D
 "cn=admin,dc=example,dc=com" \
  -w password
dn: uid=user.2004,ou=People,dc=example,dc=com
changetype: modify
add: cn
```

```
cn: Sally Tea Tree
```

An error could occur if the attribute is single-valued, if the value already exists, if the value does not meet the proper syntax, or if the value does not meet the entry's objectclass requirements. Also, make sure there are no trailing spaces after the attribute value.

Adding an attribute using the language subtype

About this task

The Directory Server provides support for attributes using language subtypes. The operation must specifically match the subtype for successful operation. Any non-ASCII characters must be in UTF-8 format.

Steps

The Directory Server provides support for attributes using language subtypes. The operation must specifically match the subtype for successful operation. Any non-ASCII characters must be in UTF-8 format.

```
$ bin/ldapmodify -h server.example.com -p 389 -w password
dn: uid=user.2004,ou=People,dc=example,dc=com
changetype: modify
add: postalAddress; lang-ko
postalAddress; lang-ko:Byung-soon Kim$2020-14 Seoul
```

Adding an attribute using the binary subtype

About this task

The Directory Server provides support for attributes using binary subtypes, which are typically used for certificates or JPEG images that could be stored in an entry. The operation must specifically match the subtype for successful operation. The version directive with a value of "1" must be used for binary subtypes. Typical binary attribute types are `userCertificate` and `jpegPhoto`.

Steps

- Use the **`ldapmodify`** command to add an attribute with a binary subtype. The attribute points to the file path of the certificate.

```
$ bin/ldapmodify -h server.example.com -p 389 -D
 "cn=admin,dc=example,dc=com" \
  -w password
version: 1
dn: uid=user.2004,ou=People,dc=example,dc=com
changetype: modify
add: userCertificate;binary
userCertificate;binary:<file:///path/to/cert
```

Deleting an attribute

Steps

Use **`ldapmodify`** with the LDIF `delete` directive to delete an attribute.

```
$ bin/ldapmodify -h server.example.com -p 389 -D "cn=admin,dc=example,dc=com"
 \
  -w password
dn: uid=user.2004,ou=People,dc=example,dc=com
changetype: modify
```

```
delete: employeeNumber
```

Deleting one value from an attribute with multiple values

About this task

You can use the LDIF `delete` directive to delete a specific attribute value from an attribute. For this example, assuming you have multiple values of `cn` in an entry (e.g., `cn: Sally Tree`, `cn: Sally Tea Tree`).

Steps

- Use **`ldapmodify`** to delete a specific attribute of a multi-valued pair, then specify the attribute pair that you want to delete. In this example, we keep `cn:Sally Tree` and delete the `cn: Sally Tea Tree`.

```
$ bin/ldapmodify -h server.example.com -p 389 -D
 "cn=admin,dc=example,dc=com" \
  -w password
dn: uid=user.2004,ou=People,dc=example,dc=com
changetype: modify
delete: cn
cn: Sally Tea Tree
```

Renaming an entry

About this task

Renaming an entry involves changing the relative distinguished name (RDN) of an entry. You cannot rename a RDN if it has children entries as this violates the LDAP protocol.

Steps

- Use the **`ldapmodify`** tool to rename an entry. The following command changes `uid=user.14` to `uid=user.2014` and uses the changetype, `newrdn`, and `deleteoldrdn` directives.

```
$ bin/ldapmodify
dn: uid=user.14,ou=People,dc=example,dc=com
changetype:moddn
newrdn: uid=user.2014
deleteoldrdn: 1
```

Moving an entry within a Directory Server

About this task

You can use **`ldapmodify`** to move an entry from one base DN to another base DN. Before running the **`ldapmodify`** command, you must assign access control instructions (ACIs) on the parent entries. The source parent entry must have an ACI that allows export operations: `allow(export)`. The target parent entry must have an ACI that allows import operations: `allow(import)`. For more information on access control instructions, see *Working with Access Control*.

Steps

- Use the **`ldapmodify`** command to move an entry from the Contractor branch to the `ou=People` branch.

```
$ bin/ldapmodify
dn: uid=user.14,ou=contractors,dc=example,dc=com
changetype:moddn
newrdn: uid=user.2014
```

```
deleteoldrdn: 0
newsuperior: ou=People,dc=example,dc=com
```

Moving an entry from one machine to another

About this task

The Directory Server provides a tool, **move-subtree**, to move a subtree or one entry on one machine to another. The subtree or entry must exist on the source server and must not be present on the target server. The source server must also support the 'real attributes only' request control. The target server must support the Ignore NO-USER-MODIFICATION request control.

ⓘ **Note:** The **move-subtree** tool moves a subtree or multiple entries from one machine to another. The tool does not copy the entries. Once the entries are moved, they are no longer present on the source server.

Steps

- Use the **move-subtree** tool to move an entry (e.g., `uid=test.user,ou=People,dc=example,dc=com`) from the source host to the target host.

```
$ bin/move-subtree --sourceHost source.example.com --sourcePort 389 \
  --sourceBindDN "uid=admin,dc=example,dc=com" --sourceBindPassword password
 \
  --targetHost target.example.com --targetPort 389 \
  --targetBindDN "uid=admin,dc=example,dc=com" --targetBindPassword password
 \
  --entryDN uid=test.user,ou=People,dc=example,dc=com
```

Moving multiple entries from one machine to another

About this task

The **move-subtree** tool provides the ability to move multiple entries listed in a DN file from one machine to another. Empty lines and lines beginning with the octothorpe character (#) will be ignored. Entry DNs may optionally be prefixed with `dn:` , but long DNs cannot be wrapped across multiple lines.

Steps

1. Open a text file, enter a list of DNs, one DN per line, and then save the file. You can also use the **ldapsearch** command with the special character "1.1" to create a file containing a list of DNs that you want to move. The following example searches for all entries that match (`department=Engineering`) and returns only the DNs that match the criteria. The results are re-directed to an output file, `test-dns.ldif`:

```
$ bin/ldapsearch --baseDN dc=example,dc=com \
  --searchScope sub "(department=Engineering)" "1.1" > test-dns.ldif
```

2. Run the **move-subtree** tool with the `--entryDNFile` option to specify the file of DNs that will be moved from one machine to another.

```
$ bin/move-subtree --sourceHost source.example.com --sourcePort 389 \
  --sourceBindDN "uid=admin,dc=example,dc=com" --sourceBindPassword password
 \
  --targetHost target.example.com --targetPort 389 \
  --targetBindDN "uid=admin,dc=example,dc=com" --targetBindPassword password
 \
  --entryDNFile /path/to/file/test-dns.ldif
```

3. If an error occurs with one of the DNs in the file, the output message shows the error. The **move-subtree** tool will continuing processing the remaining DNs in the file.

```
An error occurred while communicating with the target server: The entry
uid=user.2,ou=People,dc=example,dc=com cannot be added because an entry with
 that name
already exists
Entry uid=user.3,ou=People,dc=example,dc=com was successfully moved from
source.example.com:389 to target.example.com:389
Entry uid=user.4,ou=People,dc=example,dc=com was successfully moved from
source.example.com:389 to target.example.com:389
```

Working with the parallel-update tool

The PingDirectory Server provides a **parallel-update** tool, which reads change information (add, delete, modify, and modify DN) from an LDIF file and applies the changes in parallel. This tool is a multi-threaded version of the **ldapmodify** tool that is designed to process a large number of changes as quickly as possible.

The **parallel-update** tool provides logic to prevent conflicts resulting from concurrent operations targeting the same entry or concurrent operations involving hierarchically-dependent entries (for example, modifying an entry after it has been added, or adding a child after its parent). The tool also has a retry capability that can help ensure that operations are ultimately successful even when interdependent operations are not present in the correct order in the LDIF file (for example, the change to add a parent entry is provided later in the LDIF file than a change to add a child entry).

After the tool has applied the changes and reaches the end of the LDIF file, it automatically displays the update statistics described in the following table

Parallel-Update Tool Result Statistics

Processing Statistic	Description
Attempts	Number of update attempts
Successes	Number of successful update attempts
Rejects	Number of rejected updates
ToRetry	Number of updates that will be retried
AvgOps/S	Average operations per second
RctOps/S	Recent operations per second. Total number of operations from the last interval of change updates.
AvgDurMS	Average duration in milliseconds
RctDurMS	Recent duration in milliseconds. Total duration from the last interval of change updates.

Running the parallel-update tool

Steps

1. Create an LDIF file with your changes. The third change will generate a rejected entry because its `userPassword` attribute contains an encoded value, which is not allowed.

```
dn:uid=user.2,ou=People,dc=example,dc=com
changetype: delete

dn:uid=user.99,ou=People,dc=example,dc=com
```

```
changetype: moddn
newrdn: uid=user.100
deleteoldrdn: 1

dn:uid=user.101,ou=People,dc=example,dc=com
changetype: add
objectClass: person
objectClass: inetOrgPerson
objectClass: organizationalPerson
objectClass: top
postalAddress: Ziggy Zad$15172 Monroe Street$Salt Lake City, MI 49843
postalCode: 49843
description: This is the description for Ziggy Zad.
uid: user.101
userPassword: {SSHA}IK57iPozIQybmIJMMdRQOpIRudIDn2RcF6bDMg==

dn:uid=user.100,ou=People,dc=example,dc=com
changetype: modify
replace: st
st: TX
-
replace: employeeNumber
employeeNumber: 100
```

2. Use **`parallel-update`** to apply the changes in the LDIF file to a target server. In this example, we use ten concurrent threads. The optimal number of threads depends on your underlying system. The `--ldifFile` and `--rejectFile` options are also required.

```
$ bin/parallel-update --hostname 127.0.0.1 \
  --ldifFile changes.ldif --rejectFile reject.ldif --numThreads 10
```

```
Reached the end of the LDIF file
Attempts Successes Rejects ToRetry AvgOps/S RctOps/S AvgDurMS RctDurMS
-------- --------- ------- ------- -------- -------- -------- ---------
       4         3       1       0        3        3       26       26
All processing complete Attempted 4 operations in 1 seconds
```

3. View the rejects file for any failed updates.

```
# ResultCode=53, Diagnostic Message=Pre-encoded passwords are not allowed
 for
# the password attribute userPassword
dn: uid=user.101,ou=People,dc=example,dc=com
changetype: add
objectClass: person
objectClass: inetOrgPerson
objectClass: organizationalPerson
objectClass: top
postalAddress: Ziggy Zad$15172 Monroe Street$Salt Lake City, MI 49843
postalCode: 49843
description: This is the description for Ziggy Zad.
uid: user.101
userPassword: {SSHA}IK57iPozIQybmIJMMdRQOpIRudIDn2RcF6bDMg==
```

Working with the Watch-Entry Tool

About this task

The PingDirectory Server provides a **`watch-entry`** tool, which demonstrates replication or synchronization latency by watching an LDAP entry for changes. If the entry changes, the background of modified attributes will temporarily be red. Attributes can also be directly modified with this tool as well.

To run the **`watch-entry`** tool:

Steps

- Perform the following to connect to **server.example.com** as **uid=admin,dc=example,dc=com** and watch entry **uid=kate,ou=people,dc=example,dc=com** for changes:

```
$ bin/watch-entry --hostname server.example.com --port 389 \
  --bindDN uid=admin,dc=example,dc=com --bindPassword password \
  --entryDN uid=user,ou=people,dc=example,dc=com
```

Working with LDAP transactions

The PingDirectory Server provides support for *batched transactions*, which are processed all at once at commit time. Applications developed to perform batched transactions should include as few operations in the transaction as possible. The changes are not actually processed until the commit request is received. Therefore, the client cannot know whether the changes will be successful until commit time. If any of the operations fail, then the entire set of operations fails.

Batched transactions are write operations (add, delete, modify, modify DN, and password modify) that are processed as a single atomic unit when the commit request is received. If an abort request is received or an error occurs during the commit request, the changes are rolled back. The batched transaction mechanism supports the standard LDAP transaction implementation based on RFC 5805. It is not currently possible to process a transaction that requires changes to be processed across multiple servers or multiple Directory Server backends.

Directory servers may limit the set of controls that are available for use in requests that are part of a transaction. RFC 5805 section 4 indicates that the following controls may be used in conjunction with the transaction specification request control: assertion request control, manageDsaIT request control, pre-read request control, and post-read request control. The proxied authorization v1 and v2 controls cannot be included in requests that are part of a transaction, but they can be included in the start transaction request to indicate that all operations within the transaction should be processed with the specified authorization identity.

The PingDirectory Server supports the following additional controls in conjunction with operations included in a transaction: account usable request control, hard delete request control, intermediate client request control, password policy request control, replication repair request control, soft delete request control, soft deleted entry access request control, subtree delete request control, and undelete request control.

Requesting a batched transaction using ldapmodify

Steps

1. Use the **ldapmodify** tool's `--useTransaction` option. It provides a mechanism for processing multiple operations as part of a single batched transaction. Create a batch text file with the changes that you want to apply as a single atomic unit:

```
dn:uid=user.3,ou=People,dc=example,dc=com
changetype: delete
dn:uid=user.1,ou=People,dc=example,dc=com
changetype: modify
replace: pager
pager: +1 383 288 1090
```

2. Use **ldapmodify** with the `--useTransaction` and `--filename` options to run the batched transaction.

```
$ bin/ldapmodify --useTransaction --filename test.ldif
```

```
#Successfully created a transaction with transaction ID 400
#Processing DELETE request for uid=user.3,ou=People,dc=example,dc=com
#DELETE operation successful for DN uid=user.3,ou=People,dc=example,dc=com
#This operation will be processed as part of transaction 400
```

```
#Processing MODIFY request for uid=user.1,ou=People,dc=example,dc=com
#MODIFY operation successful for DN uid=user.1,ou=People,dc=example,dc=com
#This operation will be processed as part of transaction 400
#Successfully committed transaction 400
```

Working with Virtual Attributes

The PingDirectory Server provides mechanisms to support virtual attributes in an entry. Virtual attributes are abstract, dynamically generated attributes that are invoked through an LDAP operation, such as **`ldapsearch`**, but are not stored in the Directory Server backend. While most virtual attributes are operational attributes, providing processing-related information that the server requires, the virtual attribute subsystem allows you to create user-defined virtual attributes to suit your directory server requirements.

Overview of virtual attributes

The PingDirectory Server allows its entries to hold virtual attributes. Virtual attributes are dynamically generated attributes that are invoked through an LDAP operation, such as **`ldapsearch`**, but are not stored in the Directory Server backend. Most virtual attributes are operational attributes, providing processing-related information that the server requires. However, the virtual attribute subsystem allows you to create user-defined virtual attributes to suit your requirements.

Viewing the list of default virtual attributes

The Directory Server has a default set of virtual attributes that can be viewed using the **`dsconfig`** tool. Some virtual attributes are enabled by default and are useful for most applications. You can easily enable or disable each virtual attribute using the **`dsconfig`** tool.

The default set of virtual attributes are described in the table below. You can enable or disable these attributes using the **`dsconfig`** tool.

Virtual Attributes

Virtual Attributes	Description
ds-entry-checksum	Generates a simple checksum of an entry's contents, which can be used with an LDAP assertion control to ensure that the entry has not been modified since it was last retrieved.
ds-instance-name	Generates the name of the Directory Server instance from which the associated entry was read. This virtual attribute can be useful in load-balancing environments to determine the instance from which an entry was retrieved.
ds-pwp-state-json	Generates an operational attribute whose values is a JSON object with information about a user's current password policy state.
entryDN	Generates an `entryDN` operational attribute in an entry that holds a normalized copy of the entry's current distinguished name (DN). Clients can use this attribute in search filters.
hasSubordinates	Creates an operational attribute that has a value of TRUE if the entry has subordinate entries.
isDirectMemberOf	Generates an `isDirectMemberOf` operational attribute that contains the DNs of the groups in which the user is a member. isDirectMemberOf includes only static groups in which the user is explicitly named as a member. Compare to isMemberOf.

Virtual Attributes	Description
isMemberOf	Generates an `isMemberOf` operational attribute that contains the DNs of the groups in which the user is a member. isMemberOf includes all of the following: ▪ Static groups in which the user is explicitly named as a member ▪ Dynamic groups in which the user is a member because of the criteria for that group ▪ Static groups in which the user is a member because they are a member of a nested group that is listed as a member of the static group Compare to isDirectMemberOf.
numSubordinates	Generates an operational attribute that returns the number of child entries. While there is no cost if this operational attribute is enabled, there could be a performance cost if it is requested. Note that this operational attribute only returns the number of immediate children of the node.
subschemaSubentry	A special entry that provides information in the form of operational attributes about the schema elements defined in the server. It identifies the location of the schema for that part of the tree. ▪ ldapSyntaxes - set of attribute syntaxes ▪ matchingRules - set of matching rules ▪ matchingRuleUse - set of matching rule uses ▪ attributeTypes - set of attribute types ▪ objectClasses - set of object classes ▪ nameForms - set of name forms ▪ dITContentRules - set of DIT content rules ▪ dITStructureRules - set of DIT structure rules
User Defined Virtual Attribute	Generates virtual attributes with user-defined values in entries that match the criteria defined in the plugin's configuration. User-defined virtual attributes are intended to specify a hard-coded value for entries matching a given set of criteria.
Virtual Static Member	Generates a member attribute whose values are the DNs of the members of a specified virtual static group. Virtual static groups are best used in client applications with a large number of entries that can only support static groups and obtains all of its membership from a dynamic group. Do not modify the filter in the `Virtual Static Member` attribute as it is an advanced property and modifying it can lead to undesirable side effects.
Virtual Static Uniquemember	Generates a `uniqueMember` attribute whose values are the DNs of the members of a specified virtual static group. Virtual static groups are best used in client applications with a large number of entries that can only support static groups and obtains all of its membership from a dynamic group. Do not modify the filter in the `Virtual Static Uniquemember` attribute as it is an advanced property and modifying it can lead to undesirable side effects.

Viewing the list of default virtual attributes using dsconfig non-interactive mode

Steps

- Use **`dsconfig`** to view the virtual attributes.

```
$ bin/dsconfig list-virtual-attributes
```

Viewing virtual attribute properties

About this task

Each virtual attribute has basic properties that you can view using the **dsconfig** tool. The complete list of properties is described in the *PingDirectory Server Configuration Reference*. Some basic properties are as follows:

- **Description**. A description of the virtual attribute.
- **Enabled**. Specifies whether the virtual attribute is enabled for use.
- **Base-DN**. Specifies the base DNs for the branches containing entries that are eligible to use this virtual attribute. If no values are given, the server generates virtual attributes anywhere in the server.
- **Group-DN**. Specifies the DNs of the groups whose members can use this virtual attribute. If no values are given, the group membership is not taken into account when generating the virtual attribute. If one or more group DNs are specified, then only members of those groups are allowed to have the virtual attribute.
- **Filter**. Specifies the filters that the server applies to entries to determine if they require virtual attributes. If no values are given, then any entry is eligible to have a virtual attribute value generated.

To view virtual attribute properties:

Steps

- Use **dsconfig** to view the properties of a virtual attribute.

```
$ bin/dsconfig get-virtual-attribute-prop --name isMemberOf
```

Enabling a virtual attribute

You can enable a virtual attribute using the **dsconfig** tool. If you are using **dsconfig** in interactive command-line mode, you can access the virtual attribute menu on the Standard object menu.

Enabling a virtual attribute using dsconfig interactive mode

Steps

1. Use **dsconfig** to enable a virtual attribute. Specify the connection port, bind DN, password, and host information. Then type the LDAP connection parameter for your Directory Server: 1 for LDAP, 2 for SSL, 3 for StartTLS.

```
bin/dsconfig
```

2. On the Directory Server main menu, type `o` to change the object menu, and then type the number corresponding to **Standard**.
3. On the Directory Server main menu, type the number corresponding to virtual attributes.
4. On the **Virtual Attribute management** menu, type the number to view and edit an existing virtual attribute.
5. From the list of existing virtual attributes on the system, select the virtual attribute to work with. For this example, type the number corresponding to the `numSubordinates` virtual attribute.
6. On the **numSubordinates Virtual Attribute Properties** menu, type the number to enable the virtual attribute. On the **Enabled Property** menu for the `numSubordinates` virtual attribute, type the number to change the value to TRUE.
7. On the **numSubordinates Virtual Attribute Properties** menu, type `f` to apply the changes.

8. Verify that the virtual attribute is enabled. Note that this example assumes you have configured the group entries.

```
$ bin/ldapsearch --baseDN dc=example,dc=com "(ou=People)" numSubordinates
```

```
dn: ou=People,dc=example,dc=com
numSubordinates: 1000
```

Enabling a virtual attribute using dsconfig non-interactive mode

Steps

- Use **dsconfig** to enable the `numSubordinates` virtual attribute.

```
$ bin/dsconfig set-virtual-attribute-prop \
  --name numSubordinates --set enabled:true
```

Creating user-defined virtual attributes

User-defined virtual attributes allow you to specify an explicit value to use for the virtual attribute. There are no restrictions on the length of the value for a user-defined virtual attribute. You must only ensure that the new virtual attribute conforms to your schema, otherwise you will see an error message when you configure it.

You can define your virtual attributes using the **dsconfig** tool on the Standard object menu. Only the value property is specific to the user-defined virtual attribute. All the other properties are common across all kinds of virtual attributes, which include the following:

- **enabled** -- Indicates whether the virtual attribute should be used.
- **attribute-type** -- The attribute type for the virtual attribute that will be generated.
- **base-dn**, **group-dn**, **filter** -- May be used to select which entries are eligible to contain the virtual attribute.
- **client-connection-policy** -- May be used to configure who can see the virtual values.
- **conflict-behavior** -- Used to indicate how the server should behave if there are one or more real values for the same attribute type in the same entry. The server can either return only the real value(s), only the virtual value(s), or merge both real and virtual values.
- **require-explicit-request-by-name** -- Used to indicate whether the server should only generate values for the virtual attribute if it was included in the list of requested attributes.
- **multiple-virtual-attribute-evaluation-order-index**, **multiple-virtual-attribute-merge-behavior** -- Used to control the behavior the server should exhibit if multiple virtual attributes may be used to contribute values to the same attribute.

Creating a user-defined virtual attribute in interactive mode

About this task

The following example shows how to create a user-defined virtual attribute that assigns an Employee Password Policy to any entry that matches the filter `"(employeeType=employee)"`.

Steps

1. Run **dsconfig** to configure the user-defined virtual attribute. Specify the connection port, bind DN, password, and host information. Then type the LDAP connection parameter for your Directory Server: 1 for LDAP, 2 for SSL, 3 for StartTLS.
2. On the Directory Server main menu, type `o` to change the object menu, and then type the number to select Standard.
3. On the Directory Server main menu, type the number corresponding to virtual attributes.
4. On the **Virtual Attribute management** menu, type the number to create a new virtual attribute.

5. Next, you can use an existing virtual attribute as a template for your new attribute, or your can create a new attribute from scratch. In this example, type `n` to create a new Virtual Attribute from scratch.
6. On the **Virtual Attribute Type** menu, enter a number corresponding to the type of virtual attribute that you want to create. In this example, type the number corresponding to User Defined Virtual Attribute.
7. Next, enter a name for the new virtual attribute. In this example, enter "`Employee Password Policy Assignment`."
8. On the **Enabled Property** menu, enter the number to set the property to true (enable).
9. On the **Attribute-Type Property** menu, type the `attribute-type` property for the new virtual attribute. You can enter the OID number or attribute name. The `attribute-type` property must conform to your schema. For this example, type "`ds-pwp-password-policy-dn`".
10. Enter the value for the virtual attribute, and then press Enter or Return to continue. In this example, enter `cn=Employee Password Policy,cn=Password Policies,cn=config`, and then type Enter or Return to continue.
11. On the **User Defined Virtual Attributes** menu, enter a description for the virtual attribute. Though optional, this step is useful if you plan to create a lot of virtual attributes. Enter the option to change the value, and then type a description of the virtual attribute. In this example, enter: `Virtual attribute that assigns the Employee Password Policy to all entries that match (employeeType=employee).`
12. On the **User Defined Virtual Attribute** menu, type the number corresponding to the filter.
13. On the **Filter Property** menu, enter the option to add one or more filter properties, type the filter, and then press **Enter** to continue. In this example, type `(employeeType=employee)`. Press the number to use the filter value entered.
14. On the **User Defined Virtual Attribute** menu, type `f` to finish creating the virtual attribute.
15. Verify that the attribute was created successfully. Add the `employeeType=employee` attribute to an entry (e.g., `uid=user.0`) using **`ldapmodify`**. Add the `employeeType=contractor` attribute to another entry (e.g., `uid=user.1`).
16. Use **`ldapsearch`** to search for the user with the `employeeType=employee` attribute (e.g., `uid=user.0`). You will notice the `ds-pwp-password-policy-dn` attribute has the assigned password policy as its value.

```
$ bin/ldapsearch --baseDN dc=example,dc=com "(uid=user.0)" \
  ds-password-policy-dn

dn: uid=user.0,ou=People,dc=example,dc=com
ds-pwp-password-policy-dn: cn=Employee Password Policy,cn=Password
 Policies,cn=config
```

17. Run **`ldapsearch`** again using the filter "`(uid=user.1)`", the `ds-pwp-password-policy-dn` attribute will not be present in the entry, because the entry has the attribute, `employeeType=contractor`.

```
$ bin/ldapsearch --baseDN dc=example,dc=com "(uid=user.1)" \
  ds-password-policy-dn

dn: uid=user.1,ou=People,dc=example,dc=com
```

Creating a user-defined virtual attribute using dsconfig in non-interactive mode

Steps

- You can also use **`dsconfig`** in non-interactive command-line mode to create a virtual attribute. The following command sets up the Employee Password Policy Assignment virtual attribute introduced in the previous section:

```
$ bin/dsconfig create-virtual-attribute \
  --name "Employee Password Policy Assignment" \
```

```
  --type user-defined \
  --set enabled:true \
  --set attribute-type:ds-pwp-password-policy-dn \
  --set "filter:(employeeType=employee)" \
  --set "value:cn=Employee Password Policy,cn=Password Policies,cn=config"
```

Creating mirror virtual attributes

The PingDirectory Server provides a feature to mirror the value of another attribute in the same entry or mirror the value of the same or a different attribute in an entry referenced by the original entry. For example, given a DIT where users have a manager attributed with a value of the DN of the employee as follows:

```
dn: uid=apeters,ou=people,dc=example,dc=com
objectClass: person
objectClass: inetOrgPerson
objectClass: organizationalPerson
objectClass: top
manager:uid=jdoe,ou=people,dc=example,dc=com
uid: apeters
... (more attributes) ...
```

You can set up a mirror virtual attribute, so that the returned value for the `managerName` virtual attribute can be the `cn` value of the entry referenced by the `manager` attribute as follows:

```
$ bin/ldapsearch --baseDN dc=example,dc=com "(uid=apeters)" \
dn: uid=apeters,ou=people,dc=example,dc=com

objectClass: person
objectClass: inetOrgPerson
objectClass: organizationalPerson
objectClass: top
manager:uid=jdoe,ou=people,dc=example,dc=com
managerName: John Doe
uid: apeters
... (more attributes not shown) ...
```

Creating a mirror virtual attribute using dsconfig in non-interactive mode

About this task

You can also use **`dsconfig`** in non-interactive command-line mode to create a mirror virtual attribute. The following example sets up the `managerName` virtual attribute introduced in the previous section:

Steps

1. Update the schema to define the `managerName` attribute. In a text editor, create a file with the following schema definition for the attribute and save it as `98-myschema.ldif`, for example, in the `<server-root>/config/schema` folder. You can optionally add the attribute to an object class.

```
dn: cn=schema
objectClass: top
objectClass: ldapSubent
ry
objectClass: subschema attributeTypes: ( 1.3.6.1.4.1.32473.3.1.9.4 NAME
 'managerName'
  EQUALITY caseIgnoreMatch SUBSTR caseIgnoreSubstringsMatch SYNTAX
 1.3.6.1.4.1.1466.115.121.1.44{256}
  X-ORIGIN 'Directory Server Example' )
```

2. Restart the Directory Server.

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

3. Use **dsconfig** to create the virtual attribute.

```
$ bin/dsconfig create-virtual-attribute \
  --name "managerName" \
  --type mirror \
  --set "description:managerName from manager cn" \
  --set enabled:true \
  --set attribute-type:managerName \
  --set source-attribute:cn \
  --set source-entry-dn-attribute:manager
```

4. Verify the mirror virtual attribute by searching for an entry.

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

```
dn: uid=apeters,ou=People,dc=example,dc=com
... (attributes) ...
manager: uid=jdoe,ou=People,dc=example,dc=com
managerName: John Doe
```

Editing a virtual attribute

You can edit virtual attributes by using the **dsconfig** tool. Ensure that the virtual attribute conforms to your plugin schema. Otherwise, an error message appears when you attempt to edit the virtual attribute. If you are using **dsconfig** in interactive command-line mode, you can access the virtual attribute menu on the Standard object menu.

Editing a virtual attribute using dsconfig in non-interactive mode

Steps

- Use **dsconfig** to change a property's value.

```
$ bin/dsconfig set-virtual-attribute-prop --name dept-number \
  --set "value:111"
```

Deleting a virtual attribute

About this task

You can delete virtual attributes using the **dsconfig** tool. If you are using `dsconfig` in interactive command-line mode, you can access the virtual attribute menu on the Standard object menu.

To delete a virtual attribute:

Steps

- Use **dsconfig** to delete an existing virtual attribute.

```
$ bin/dsconfig delete-virtual-attribute --name dept-number
```

Working with Composed Attributes

PingDirectory Server's support for virtual attributes is a useful facility that can dynamically generate values in entries for a wide range of use cases. These are used to generate key operational attributes like entryDN, subschemaSubentry, isMemberOf, numSubordinates, hasSubordinates, and ds-entry-

checksum. The values of other attributes in the same or different entries can be mirrored, and values can be constructed from other attributes in the same entry. Items such as password policies and privileges can be dynamically, and it is also possible to create custom virtual attribute providers with the Server SDK using a wide variety of logic.

However, virtual attributes are not an ideal solution for all use cases. This section provides information on conposed attributes, which dynamically generate attribute values to complement virtual attributes. Composed attributes are suited for some scenarios that virtual attributes do not support.

Virtual attribute limitations

Performance limitations

There may be performance limitations associated with generating virtual attributes. Although the performance impact of actually generating the values is limited by only computing these values if they are to be returned to the client, there can still be a noticeable impact on performance. In addition, virtual attributes may be included or excluded based on a range of criteria and the evaluation of this criteria (for example, based on group membership) may impact performance..

Indexing limitations

Virtual attributes do not mix well with indexing. You cannot index the values of a virtual attribute since these values can change from one invocation to the next.

Some virtual attributes do implement support for an index-like functionality in that they offer logic that may be able to quickly identify entries that have specified virtual attribute values. These include the following:

- When mirroring values from another attribute in the same entry, and when that source attribute type is indexed, the mirror virtual attribute provider can use an internal search to identify matching entries.
- The entryDN virtual attribute provider can retrieve the entry with the specified DN.
- The isMemberOf virtual attribute provider can use the server's group manager to identify the members of a targeted group.

However, this is not an option for all types of virtual attributes. For example, it is not available for user-defined virtual attributes that have a static value, for constructed virtual attributes whose values are composed from multiple other attributes in the same entry, for mirror virtual attributes whose values come from other entries, or for any custom virtual attribute types implemented using the Server SDK.

In addition, there is not a good solution for the case where you want to have a mix of real and virtual values for the same attribute type (whether in the same entry or in different entries). If you define an index for that attribute type, it will only include entries with real values; entries with virtual values will be overlooked. On the other hand, if there is no index but the virtual attribute provider does support some search functionality, only entries with virtual values will be matched and entries with only real values will be excluded.

Unexpected behavior for write operations

Another significant issue with virtual attributes is that they can result in unexpected behaviors when targeted by write operations. These behaviors include the following:

- Attempting to delete a virtual value will fail with a "no such attribute" result.
- If the virtual attribute provider is configured with a conflict-behavior of real-overrides-virtual, then attempting to add a value to an entry that only has virtual values will cause the virtual values to disappear.
- If the virtual attribute provider is configured with a conflict-behavior of real-overrides-virtual, then attempting to remove all real values of an entry will cause the virtual values to appear.
- If the virtual attribute provider is configured with a conflict-behavior of virtual-overrides-real, then attempting to add new values or replace the set of existing values will yield a success result, but the operation will have no visible effect on the entry.

- If the virtual attribute provider is configured with a conflict-behavior of merge-real-and-virtual, then attempting to replace the set of values for an entry will yield a success result, but only the real values will have been replaced, and the virtual values will remain.

There is currently no method to prevent attempts to write to attributes with virtual values. The NO-USER-MODIFICATION constraint in attribute type definitions is honored, but this constraint can only be applied to operational attribute types; this is not an acceptable limitation in many cases. Access control restrictions could work for many clients, but will not have any effect for requesters with the bypass-acl privilege.

Overview of composed attributes

Composed attributes are generated when an entry is written. This includes entries that are created by add operations and LDIF imports; these may also be created, updated, or removed when processing modify and modify DN operations. The server may optionally permit or reject attempts to alter the generated values.

Composed values are based solely on a combination of static text and the contents of the entry in which the value is to be generated. It is not possible to construct values using content from other entries in the server, as the necessary content may not be available at the time the value is to be generated (for example, if the referenced entry does not yet exist), and it would also be difficult to ensure that the valuet is kept up to date whenever the referenced entry changes or is removed. Values are generated using the same logic that is available for constructed virtual attributes, as well as for constructed attribute mapping functionality in the Synchronization Server.

The composed attribute functionality is implemented using a plugin. The plugin is invoked during the pre-operation phases for add, modify, and modify DN operations, and it also operatse during LDIF import.

The server also provides an administrative task that can be used to iterate through all entries in a backend and generate values as necessary. This is useful when a new composed attribute plugin instance is created after the backend has already been populated and you do not wish to export the data to LDIF and re-import.

The server does not provide any first-class extensibility mechanisms for composed attributes. Unlike virtual attributes, which can be created through the Server SDK, composed attribute support is intentionally limited to what can be provided by the server. You can implement your own plugins to generate your own data if necessary.

Composed attribute plugin configuration properties

The composed attribute plugin provides the following configuration properties:

- plugin-type - The plugin type values for which the server will be registered. This is a mandatory multi-valued property with a default that includes all of the ldifimport, preoperationadd, preoperationmodify, and preoperationmodifydn values. It is recommend in general that you do not override this default set, but this may be useful in some rare corner cases. The available options include:
  - ldifimport - Indicates that values should be generated for the target attribute in any appropriate entries created during LDIF import processing.
  - preoperationadd - Indicates that values should be generated for the target attribute in any appropriate entries created by add operations.
  - preoperationmodify - Indicates that values should be generated or altered for the target attribute in any appropriate entries updated by modify operations.
  - preoperationmodifydn - Indicates that values should be generated or altered for the target attribute in any appropriate entries updated by modify DN operations.
- attribute-type - The name or OID of the attribute type for which values are to be generated. This attribute type must be defined in the schema. This is a mandatory, single-valued property with no default. If multiple composed attribute types are desired, then a separate instance of the plugin should be configured for each. However, if multiple values should be composed for the same attribute type

from different sets of source attributes, then a single instance of the plugin is recommended with multiple value patterns.

- value-pattern - The value pattern that specifies the logic that should be used when constructing values. This pattern must be compatible with the format expected by the com.unboundid.directory.server.sync.mapping.ConstructedValue class. This is a mandatory, multi-valued property in which the order of the values will be preserved. If this is configured with multiple values, then the behavior the plugin should exhibit will be controlled by the multiple-value-pattern-behavior property. Note that a value pattern will only be used for an entry if that entry contains all of the attributes referenced in the value pattern. Also note that virtual attributes are not considered when generating values.
- multiple-value-pattern-behavior - The behavior that the server should exhibit if multiple value-pattern values are configured. This will be a single-valued property with a default value of use-first-non-rejected-value-pattern-with-non-empty-values-but-may-reject.

  ⓘ **Note:** The primary reason that a rejection may occur is as a result of multivalued source attributes, but it could also occur if an unexpected error occurs while trying to generate values.

  The available options include:

  - use-first-non-rejected-value-pattern-with-non-empty-values-but-may-reject - Indicates that the server should only use values generated from the first value pattern that is not rejected and yields a non-empty result. If all value patterns yield empty results (for example, if the entry is missing at least one source attribute for each of the value patterns), then no composed value will be generated. If no value pattern yields a non-empty result but at least one is rejected, then the operation will fail, or the entry will be rejected if performing an LDIF import.
  - use-first-non-rejected-value-pattern-with-non-empty-values-and-never-reject - Indicates that the server should only use values generated from the first value pattern that is not rejected and yields a non-empty result. If none of the value patterns yield non-empty results (regardless of whether any of them is rejected), then no composed value will be generated.
  - use-first-rejection-or-first-value-pattern-with-non-empty-values - Indicates that the server should only use values generated from the first value pattern that yields non-empty results, but if a rejection is encountered before a non-empty result is obtained, then the operation (or LDIF entry) will be rejected. If none of the value patterns is not rejected and yields a non-empty result, then no composed value will be generated.
  - use-all-non-rejected-value-patterns-with-non-empty-values-but-may-reject - Indicates that the server should use all values generated by all of the value patterns that were not rejected for the entry. As long as at least one value pattern generated a non-empty result, then the entire set of values generated by all value patterns will be used. If none of the value patterns yielded non-empty results but none were rejected, then no composed value will be generated. If none of the value patterns yielded non-empty results and at least one was rejected, then the operation will fail or the LDIF entry will be rejected.
  - use-all-non-rejected-value-patterns-with-non-empty-values-and-never-reject - Indicates that the server should use all values generated by all of the value patterns that were not rejected for the entry. As long as at least one value pattern generated a non-empty result, then the entire set of values generated by all value patterns will be used. If none of the value patterns yielded non-empty results (regardless of whether any of them was rejected), then no composed value will be generated.
  - use-first-rejection-or-all-value-patterns-with-non-empty-values - Indicates that the server should use all of the values generated by all of the value patterns as long as none of them is rejected. If any value pattern is rejected, then the operation will fail or the LDIF entry will be rejected. If none of the value patterns is rejected but none of them generates any values, then no composed values will be generated.
- multi-valued-attribute-behavior - The behavior that the server should exhibit if any of the attributes used in the value pattern have multiple values. Note that at most one source attribute may have multiple values; if multiple source attributes have multiple values, and if this property is configured to try to use all of them, then the attempt to generate composed values for the entry may fail. This will be a

single-value property with a default value of use-all-values-if-possible-but-reject-if-not. Available options include:

  - use-first-value - Indicates that the server should only use the first value for each source attribute referenced in the value pattern. With this value, entries should never be rejected for having multivalued attributes.
  - reject-entries-with-any-multivalued-source-attribute - Indicates that the server should reject any entry that has more than one value for any of the source attributes.
  - use-all-values-if-possible-but-reject-if-not - Indicates that the server should use all values for the source attributes, as long as at most one of them has multiple values. If multiple source attributes have multiple values, the entry will be rejected.
  - use-all-values-if-possible-but-only-first-value-if-not - Indicates that the server should use all values for the source attributes, as long as at most one of them has multiple values. If multiple source attributes have multiple values, then only the first value from each source attribute will be used so that the entry will not be rejected.
- target-attribute-exists-during-initial-population-behavior - The behavior that the server should exhibit for add operations and LDIF imports for entries that already contain one or more values for the target attribute. This will be a single-valued property with a default of preserve-existing-values. The available options include:
  - preserve-existing-values - Indicates that the server should preserve existing values in entries that are added or imported, and should only construct values for entries that do not already contain the target attribute.
  - overwrite-existing-values - Indicates that the server should only use generated values for the target attribute and that any values provided during an add or LDIF import will be discarded.
  - merge-existing-and-composed-values - Indicates that the server should preserve any existing values for the target attribute, but should also construct any additional values.

    ⓘ **Note:** This option may only be used if the target attribute type allows multiple values.
  - reject-existing-values-in-add-but-preserve-in-ldif-import - Indicates that the server should reject any add operation that attempts to provide any values for the target attribute, but that it should preserve any existing values when performing an LDIF import. Note that the server will not reject entries that already contain the target attribute when performing an LDIF import since this would interfere with the ability to export data to LDIF from a server that contains composed values and then re-import the LDIF file.
  - reject-existing-values-in-add-but-overwrite-in-ldif-import - Indicates that the server should reject any add operation that attempts to provide any values for the target attribute, but that it should replace any existing values in an LDIF import with the values which would be composed for that entry.
  - reject-existing-values-in-add-but-merge-in-ldif-import - Indicates that the server should reject any add operation that attempts to provide any values for the target attribute, but should preserve any existing values when performing an LDIF import and add any additional values that should be composed for the entry.
- update-source-attribute-behavior - The behavior that the server should exhibit for entries that are updated by modify or modify DN operations that alter values for one of the attributes used in the value pattern. This will be a single-valued property with a default of replace-composed-values. The available options include:
  - replace-composed-values - Indicates that the server should only update values for the target attribute that match those that would have been generated by the plugin. Values that would not have been generated by the plugin will be preserved.
  - replace-all-values - Indicates that the server should always replace the target attribute with new composed values, even if the existing values were not generated by the plugin.
  - preserve-existing-values - Indicates that the server should always preserve the existing values for the target attribute, even if they were generated by the plugin.

- source-attribute-removal-behavior - The behavior that the server should exhibit when an entry is updated in a manner that removes one or more source attributes needed to compose values for the target attribute. This will be a single-valued property with a default of preserve-non-composed-values. The available options include:
  - preserve-non-composed-values - Indicates that the server will attempt to remove any values for the target attribute that would have been generated for that entry by the associated value patterns, but that it will preserve any other values that may exist for the target attribute. However, if the target attribute is required by any of the object classes in the entry and if that entry only contains composed values for the target attribute, then the composed values will be preserved to avoid violating the schema.
  - preserve-all-values - Indicates that the server will preserve any values that exist for the target attribute, regardless of whether they would have been composed by any of the value patterns.
  - remove-all-values-but-preserve-all-if-required - Indicates that the server will attempt to remove all values for the target attribute from the entry, regardless of whether they would have been composed, but that it will fall back to a behavior of preserve-all-values if the target attribute type is required by any of the entry's object classes.
  - remove-all-values-but-preserve-non-composed-values-if-required - Indicates that the server will attempt to remove all values for the target attribute from the entry, regardless of whether they would have been composed, but that it will fall back to a behavior of preserve-non-composed-values if the target attribute type is required by any of the entry's object classes.
- update-target-attribute-behavior - The behavior that the server should exhibit when a client attempts to alter the target attribute with a modify or modify DN operation. This will be a single-valued property with a default of always-allow. The aAvailable options include
  - always-allow - Indicates that the server will always permit clients to alter the value of the target attribute, regardless of whether these values were generated.
  - allow-only-for-non-composed-values - Indicates that the server will only permit clients to alter values for the target attribute that were not generated by the plugin. Attempts to alter composed values will be rejected.
  - never-allow - Indicates that the server should reject any attempt to alter the target attribute, regardless of the values it may have.
- include-base-dn - A base DN that may be used to restrict the set of entries for which values will be composed. The include/exclude criteria configuration options are described in more detail below.
- exclude-base-dn - A base DN that may be used to restrict the set of entries for which values will be composed. The include/exclude criteria configuration options are described in more detail below.
- include-filter - A search filter that may be used to restrict the set of entries for which values will be composed. The include/exclude criteria configuration options ware described in more detail below.
- exclude-filter - A search filter that may be used to restrict the set of entries for which values will be composed. The include/exclude criteria configuration options are described in more detail below.
- updated-entry-newly-matches-criteria-behavior - The behavior that the server should exhibit if an entry that previously did not satisfy either the base DN or filter criteria, but is updated so that it does match any configured base DN or filter criteria. This will be a single-valued property with a default value of preserve-existing-values-and-compose-new-values. The available options include:
  - preserve-existing-values-without-composing-new-values - Indicates that if the entry already had one or more values for the target attribute, those values will be preserved and no additional composed values will be added. If the entry did not contain the target attribute, then no composed values will be generated for the entry.
  - preserve-existing-values-or-compose-new-values - Indicates that if the entry already had one or more values for the target attribute, those values will be preserved and no additional composed values will be added. However, if the entry did not contain the target attribute, then composed values will be generated for the entry.
  - preserve-existing-values-and-compose-new-values - Indicates that if the entry already had one or more values for the target attribute, those values will be preserved and any additional values that

would be composed for that entry will also be added. If the entry did not contain the target attribute, then composed values will be generated for the entry.

  - compose-new-values-without-preserving-existing-values - Indicates that if the entry already had one more composed values for the target attribute, then those values will be removed and replaced with the composed values that would be generated for it. If the entry did not contain the target attribute, then composed values will be generated for it. If no composed values would be generated for the entry (for example, if it was missing at least one source attribute for each value pattern), then the entry will not include the target attribute, regardless of whether it contained existing values for the attribute.

- updated-entry-no-longer-matches-criteria-behavior - The behavior that the server should exhibit if an entry previously satisfied the configured base DN and filter criteria but has been updated so that it no longer matches that criteria. This will be a single-valued property with a default of preserve-all-values. The avvailable options include:

  - preserve-all-values - Indicates that the server will preserve any values that exist for the target attribute, regardless of whether they would have been composed by any of the value patterns.
  - preserve-non-composed-values - Indicates that the server will attempt to remove any values for the target attribute that would have been generated for that entry by the associated value patterns, but will preserve any other values that may exist for the target attribute. However, if the target attribute is required by any of the object classes in the entry, and if that entry only contains composed values for the target attribute, then the composed values will be preserved to avoid violating the schema.
  - remove-all-values-but-preserve-all-if-required - Indicates that the server will attempt to remove all values for the target attribute from the entry, regardless of whether they would have been composed, but that it will fall back to a behavior of preserve-all-values if the target attribute type is required by any of the entry's object classes.
  - remove-all-values-but-preserve-non-composed-values-if-required —-Indicates that the server will attempt to remove all values for the target attribute from the entry, regardless of whether they would have been composed, but that it will fall back to a behavior of preserve-non-composed-values if the target attribute type is required by any of the entry's object classes.

By default, attribute values will be composed for every entry in any public backend that contains all of the source attributes used in the appropriate value patterns. If you wish to restrict composed values to a specific set of entries, that can be done with the include-base-dn, exclude-base-dn, include-filter, and exclude-filter properties.

If the include-base-dn property is provided, then values will only be composed for entries at or below one of those base DNs. If the exclude-base-dn property is provided, then values will not be composed for any entries at or below one of those base DNs. If both properties are provided and a given entry resides below both an include and an exclude base DN, then the exclude base DN will take precedence, and values will not be generated for that entry. If neither property is specified, then the naming contexts of all public backends will be used as the default set of include base DNs.

If the include-filter property is provided, then values will only be composed for entries that match at least one of those filters. If the exclude-filter property is provided, then values will not be composed for any entries that match at least one of those filters. If both properties are provided and a given entry matches both include and exclude filters, then the exclude filter will take precedence, and values will not be generated for that entry. If neither property is specified, then the server will behave as if an include filter of "(&)" had been specified. Note that no virtual attributes will be considered when evaluating this filter, although composed values may be taken into consideration. See *Composed attribute dependency considerations* on page 541 for more information about the order in which composed attribute definitions will be evaluated.

Populate composed attribute values task

When a composed attribute plugin is created in the server, it takes effect immediately and starts creating values for any appropriate add, modify, and modify DN operations. However, existing entries will not automatically be populated with generated values. This can be achieved by exporting the data to LDIF and re-importing it, but this would require temporarily taking each instance out of service since importing

into a single instance will have no effect on the data in other instances. The LDIF import option is really only feasible when you are starting from scratch, not when you have existing data. To address this need, a “populate composed attribute values” task is provided. This task iterates through all entries in one or more backends and generates values as appropriate for any entries.

This task provides several number of configuration attributes, including the following:

- ds-task-populate-composed-attribute-plugin-config - An optional, multi-valued attribute that can be used to specify the names or DNs of the composed attribute plugin configuration entries for which values should be generated. If this attribute is omitted, then values will be generated for all defined and enabled composed attribute plugin instances.
- ds-task-populate-composed-attribute-backend-id - An optional, multi-valued attribute that may be used to specify the backend IDs of the backends in which values should be generated. If this attribute is omitted, then an appropriate set of backends will be determined based on the include and exclude base DNs for the selected plugins.
- ds-task-populate-composed-attribute-max-rate-per-second - An optional, single-valued integer attribute that can be used to specify the maximum number of entries to be updated per second. If this attribute is omitted, then no rate limit will be imposed.

The task uses the same logic that the plugin uses for entries imported from LDIF when deciding what values to include in the entry, but it will use internal modify operations to make any necessary updates to the entry. These changes will be replicated to other servers in the topology, so the task only needs to be invoked on a single instance. However, it is recommended running it only after the composed attribute plugins have been configured on all instances in the topology, as this will ensure that no entries (including those added or updated while the task is running) are overlooked.

A populate-composed-attribute-values command-line tool that can be used to invoke the task and monitor its progress is provided. Since this processing is not expected to be needed on a regular basis, this is not offered as a recurring task.

Composed attribute dependency considerations

Composed attributes must not have any dependency on virtual attributes. The server will not use virtual attribute values when generating values from a value pattern, and it will also not use virtual attribute values when determining whether an entry matches an include or exclude filter.

There is normally no need to have one composed attribute depend on another. If one composed attribute is constructed from other attributes in the entry, then another composed attribute that needs to use another composed attribute as its source can simply include the value pattern for that first composed attribute as part of its own value pattern. However, this may not be sufficient if the composed values can be overridden by clients.

If there is a reason that one composed attribute needs to depend on another, then it should be sufficient to ensure that the following requirements are satisfied:

- Any composed attribute plugins that have dependencies should be configured so that they will be invoked in the appropriate order. This may be accomplished through the plugin-order-ldif-import, The populate composed attribute values task should be run separately for each of the attributes to be generated. The first run should be to generate values for any composed attributes that are not dependent on any other composed attributes. The second run should generate values for any composed attributes that were created during the first run. If multiple levels of nesting are required, then additional runs of the task will also be required.plugin-order-pre-operation-add, plugin-order-pre-operation-modify, and plugin-order-pre-operation-modify-dn properties in the plugin root configuration.
-

Schema validation considerations

The server only generates composed attributes for entries in which that attribute is allowed to be present. For user attributes, tist means that the target attribute type must be permitted by at least one of the

object classes associated with the entry. If an entry does not have any object class that permits the target attribute, then an attempt to generate composed values for the entry will fail with an objectClassViolation (65) result. This restriction does not exist for operational attributes.

The server allows composed attributes to satisfy mandatory attribute requirements. That is, if the target attribute is declared as a MUST type for any of the entry's object classes, then it must be possible for a client to add an entry that does not include any values for that attribute type as long as the server will compose a value for that attribute.

The server also enforces attribute syntax restrictions for composed attributes. If a composed attribute would violate the syntax for the associated attribute type, then the operation that would result in that composed attribute value isrejected with an INVALID_ATTRIBUTE_SYNTAX result. This may be overridden on a per-attribute-type basis using the permit-syntax-violations-for-attribute property in the global configuration.

ⓘ **Note:** If this option is used to permit values that violate the associated syntax, then matching operations involving malformed values may not behave in a predictable manner,

.

The server should also enforce the SINGLE-VALUE constraint for the target attribute type. If an attribute type is defined with this constraint, it is not possible to configure the composed attribute plugin in a manner that could generate multiple values for that attribute and any operation that would result in multiple values for the target attribute is rejected.

Composed attributescannot set values for operational attributes that are defined with the NO-USER-MODIFICATION constraint.

Replication considerations

The composed attribute configuration should be identical for all servers in the replication topology. Whenever a composed attribute plugin is configured for one instance, the same definition should be added to all other instances in the topology.

Composed attribute values should only be generated for the server that processes the request from the client. When that operation is replicated to other servers, the composed attribute changes should be included as part of that operation.

Replicated changes involving composed attributes should not be subject to restrictions that would prevent external clients from writing to those values. This includes composed values for operational attribute types with the NO-USER-MODIFICATION constraint as well as for cases in which the plugin forbids clients from modifying values.

ⓘ **Note:** The plugin rejects any modify DN operation that alters source attributes in a way that would result in composed attribute changes.

Synchronization Server considerations

There should not be any issues synchronizing changes involving composed attributes from a PingDirectory Server instance to some other type of endpoint. Changes that create or update composed attributes should be written to the LDAP changelog, just like changes to regular attributes.

Synchronization to a PingDirectory Server configured with one or more composed attribute plugin instances may be more problematic. In particular, it may not be possible to a composed attribute type if the plugin that generates values for that attribute is configured to prohibit external clients from altering values of that attribute.

If the PingDirectory Server is used as a sync destination in an environment in which some attribute values may need to be composed, then it may be desirable to compose those values in the Synchronization

Server using a constructed attribute mapping rather than attempting to generate them in the Directory Server with the composed attribute plugin.

Directory Proxy Server considerations

The use of PingDirectoryProxy Server is not expected to require any special consideration in an environment involving composed attributes. This should be true for both simple and entry-balanced proxy configurations.

Troubleshooting considerations

For operations that are rejected for reasons related to composed attribute processing (for example, if an add, modify, or modify DN operation is rejected because the composed value would have violated the syntax for the target attribute type, or if an attempt was made to alter a composed attribute value in a way that is not permitted by the plugin), the diagnostic message is returned to the client and included in the access log.

The composed attribute plugin also contains detailed debug logging support. If you encounter unexpected behavior when using the composed attribute plugin, you should enable debugging for the plugin to provide information to help diagnose the problem.

Security considerations

The primary security consideration for composed attributes is that they may expose the values of other attributes. For example, if the cn attribute is composed from the values of the givenName and sn attributes, then a user with permission to read the cn attribute could determine the values of the givenName and sn attributes even if they do not have permission to read these attributes directly. However, this is not usually a significant concern and you can address this by ensuring that the user's access-control configuration restricts access to source attributes used in a composed attribute value pattern and imposes similar restrictions to the composed attribute.

Limitations of composed attributes relative to virtual attributes

Composed attribute support provides an alternative to a specific set of use cases, and these use cases may also be addressed by virtual attribute support. While the composed attribute solution may provide improvements in performance, searchability, and write behavior, it also has a number of limitations relative to the virtual attribute subsystem. Some of those limitations include:

- When creating a composed attribute plugin in the server, manual action is required to populate composed values in existing entries. When enabling a virtual attribute, the attribute is immediately available to clients.
- When removing a virtual attribute from the server, values that would have been generated for the attribute will no longer be present. When removing or disabling a composed attribute plugin, composed values that had already been generated will remain.
- Virtual attributes can be configured so that the values are only generated under certain conditions (for example, only for certain clients), and so that different values may be generated for different clients. Composed attributes will be the same for all clients (although access controls can be used to restrict which clients have access to them).
- Virtual attributes do not require any additional storage or memory usage. Composed attributes actually exist in the database and therefore require additional disk space and memory for caching.
- The composed attribute plugin can only generate values from a combination of static text and values from other attributes in the same entry. Virtual attributes can use a much wider range of logic when generating values, including custom logic implemented using the Server SDK.

Encrypting Sensitive Data

The Directory Server provides several ways that you can protect sensitive information in the server. If not enabled during server setup, you can enable on-disk encryption for data in backends as well as in the changelog and the replication databases, and you can also protect sensitive attributes by limiting the ways that clients may interact with them.

This chapter presents the following topics:

About encrypting and protecting sensitive data

The Directory Server provides an encryption-settings database that holds encryption and decryption definitions to protect sensitive data. You can enable on-disk encryption for data in backends as well as in the changelog and the replication databases. You can also protect sensitive attributes by limiting the ways that clients may interact with them.

About the Encryption-Settings Database

The encryption-settings database is a repository that the server uses to hold information for encrypting and decrypting data. The database contains any number of *encryption-settings definitions* that specifies information about the cipher transformation and encapsulates the key used for encryption and decryption.

Before data encryption can be enabled, you first need to create an encryption-settings definition. An encryption-settings definition specifies the cipher transformation that should be used to encrypt the data, and encapsulates the encryption key. The `encryption-settings` command-line tool can be used to manage the encryption settings database, including creating, deleting, exporting, and importing encryption-settings definitions, listing the available definitions, and indicating which definition should be used for subsequent encryption operations.

Although the encryption-settings database can have multiple encryption-settings definitions, only one of them can be designated as the *preferred* definition. The preferred encryption-settings definition is the one that will be used for any subsequent encryption operations. Any existing data that has not yet been encrypted remains unencrypted until it is rewritten (e.g., as a result of a modify or modify DN operation, or if the data is exported to LDIF and re-imported). Similarly, if you introduce a new preferred encryption-settings definition, then any existing encrypted data will continue to use the previous definition until it is rewritten. If you do change the preferred encryption-settings definition for the server, then it is important to retain the previous definitions until you are confident that no remaining data uses those older keys.

Supported Encryption Ciphers and Transformations

The set of encryption ciphers that are supported by the Directory Server is limited to those ciphers supported by the JVM in which the server is running. For specific reference information about the algorithms and transformations available in all compliant JVM implementations, see the following:

- Java Cryptography Architecture Reference Guide
- Java Cryptography Architecture Standard Algorithm Name Documentation

When configuring encryption, the cipher to be used must be specified using a key length (in bits) and either a cipher algorithm name (e.g., "AES") or a full cipher transformation which explicitly specifies the mode and padding to use for the encryption (e.g., "AES/CBC/ PKCS5Padding"). If only a cipher algorithm is given, then the default mode and padding for that algorithm will be automatically selected.

The following cipher algorithms and key lengths have been tested using the Sun/Oracle JVM:

Cipher Algorithms

Cipher Algorithm	Key Length (bits)
AES	128
Blowfish	128
DES	64
DESede	192
RC4	128

ⓘ **Note:** By default, some JVM implementations may come with limited encryption strength, which may restrict the key lengths that can be used. For example, the Sun/Oracle JVM does not allow AES with 192-bit or 256-bit keys unless the unlimited encryption strength policy files are downloaded and installed.

The Directory Server supports four Cipher Stream Providers, which are used to obtain cipher input and output streams to read and write encrypted data.

Cipher Stream Providers

Cipher Stream Providers	Description
Default	Default cipher stream provider using a hard-coded default key.
File-Based	Used to read a specified file in order to obtain a password used to generate cipher streams for reading and writing encrypted data.
Third-Party	Used to provide cipher stream provider implementations created in third-party code using the Server SDK.
Wait-for-Passphrase	Causes the server to wait for an administrator to enter a passphrase that will be used to derive the key for cipher streams. You can supply the passphrase to the server by running **`encryption-settings supply-passphrase`**.

Using the encryptions-settings Tool

About this task

The **`encryption-settings`** tool provides a mechanism for interacting with the server's encryption-settings database. It may be used to list the available definitions, create new definitions, delete existing definitions, and indicate which definition should be the preferred definition. It may also be used to export definitions to a file for backup purposes and to allow them to be imported for use in other Directory Server instances.

To list the available encryption definitions:

Steps

- Use the **`encryption-settings`** tool with the **`list`** subcommand to display the set of available encryption settings definitions. This subcommand does not take any arguments. For each definition, it will include the unique identifier for the definition, as well as the cipher transformation and key length that will be used for encryption and whether it is the preferred definition.

```
$ bin/encryption-settings list
```

```
Encryption Settings Definition ID: 4D86C7922F71BB57B8B5695D2993059A26B8FC01
Preferred for New Encryption: false
```

```
Cipher Transformation: DESede
Key Length (bits): 192

Encryption Settings Definition ID: F635E109A8549651025D01D9A6A90F7C9017C66D
Preferred for New Encryption: true
Cipher Transformation: AES
Key Length (bits): 128
```

Creating encryption-settings definitions

About this task

To create a new encryption-settings definition, use the `create` subcommand. This subcommand takes the following arguments:

- **--cipher-algorithm {algorithm}**. Specifies the base cipher algorithm that should be used. This should just be the name of the algorithm (e.g., "AES", "DES", "DESede", "Blowfish", "RC4", etc.). This argument is required.
- **--cipher-transformation {transformation}**. Specifies the full cipher transformation that should be used, including the cipher mode and padding algorithms (e.g., "AES/CBC/ PKCS5Padding"). This argument is optional, and if it is not provided, then the JVM-default transformation will be used for the specified cipher algorithm.
- **--key-length-bits {length}**. Specifies the length of the encryption key in bits (e.g., 128). This argument is required.
- **--set-preferred**. Indicates that the new encryption-settings definition should be made the preferred definition and therefore should be used for subsequent encryption operations in the server. When creating the first definition in the encryption-settings database, it will automatically be made the preferred definition.

To create an encryption-settings definition:

Steps

- Use the **`encryption-settings`** tool with the `create` subcommand to specify the definition.

```
$ bin/encryption-settings create --cipher-algorithm AES \
  --key-length-bits 128 --set-preferred

Successfully created a new encryption settings definition with ID
F635E109A8549651025D01D9A6A90F7C9017C66D
```

Changing the preferred encryption-settings definition

About this task

To change the preferred encryption-settings definition, use the **`encryption-settings`** tool with the `set-preferred` subcommand. This subcommand takes the following arguments:

- **--id {id}**. Specifies the ID for the encryption-settings definition to be exported. This argument is required.

Steps

- Use the **`encryption-settings`** tool with the `set-preferred` subcommand to change a definition to a preferred definition.

```
$ bin/encryption-settings set-preferred --id
 4D86C7922F71BB57B8B5695D2993059A26B8FC01

Encryption settings definition 4D86C7922F71BB57B8B5695D2993059A26B8FC01 was
```

```
successfully set as the preferred definition for subsequent encryption
 operations
```

Deleting an encryption-settings definition

About this task

To delete an encryption-settings definition, use the **`encryption-settings`** tool with the `delete` subcommand. The subcommand takes the following arguments:

- **--id {id}**. Specifies the ID for the encryption-settings definition to be deleted. This argument is required.

Never delete an encryption-settings definition if data in the server is still encrypted using the settings contained in that definition. Any data still encrypted with a definition that has been removed from the database will be inaccessible to the server and will cause errors for any attempt to access it. This includes the replicationChanges and changelog databases, which the **`re-encode-entries`** tool will not re-encode with the new encryption-settings definition. Therefore, wait for the amount of time defined in the **`replication-purge-delay`**, of the Replication Server, and **`changelog-maximum-age`** of the changelog Backend (if enabled) before removing previous encryption-settings definitions. To safely delete a compromised encryption-settings definition, see the *Dealing with a Compromised Encryption Key* section.

To stop using a definition for encryption and use a different definition, make sure that the desired definition exists in the encryption-settings database and set it to be the preferred definition. As long as the encryption key has not been compromised, there is no harm in having old encryption-settings definitions available to the server, and it is recommended that they be retained just in case they are referenced by something.

The preferred encryption-settings definition cannot be deleted unless it is the only one left. To delete the currently-preferred definition when one or more other definitions are available, make one of the other definitions preferred as described in the previous section.

Steps

- Use the **`encryption-settings`** command with the `delete` subcommand. Make sure to include the `--id` argument to specify the definition.

```
$ bin/encryption-settings delete --id
 F635E109A8549651025D01D9A6A90F7C9017C66D
```

```
Successfully deleted encryption settings definition
F635E109A8549651025D01D9A6A90F7C9017C66D
```

Configuring the encryption-settings database

About this task

Because the encryption-settings database contains the encryption keys used to protect server data, the contents of the encryption-settings database is itself encrypted. By default, the server will derive a key to use for this purpose, but it is recommended that you customize the logic used to access the encryption-settings database with a cipher stream provider. The Server SDK provides an API that can be used to create custom cipher stream provider implementations, but the server also comes with one that will obtain the key from a PIN file that you create (see the example procedure below).

Steps

1. Use **`dsconfig`** to configure the server so that the encryption-settings database is encrypted with a PIN contained in the file `config/encryption-settings.pin`.

```
$ bin/dsconfig create-cipher-stream-provider \
  --provider-name "Encryption Settings PIN File" \
```

```
  --type file-based \
  --set enabled:true \
  --set password-file:config/encryption-settings.pin
```

2. Use **`dsconfig`** to set the global configuration property for the cipher stream provider, which sets the on-disk encryption.

```
$ bin/dsconfig set-global-configuration-prop \
  --set "encryption-settings-cipher-stream-provider:Encryption Settings PIN
 File"
```

3. Use the `encryption-settings` tool to create a new encryption-settings definition. This command automatically generates a new 256-bit encryption key for use with AES encryption, and mark it as the preferred definition for future encryption operations in the server. Note that this command will fail if you do not have the unlimited encryption strength policy installed as described in the previous section (if you do not have that policy installed, then you are restricted to a 128-bit key for AES encryption).

```
$ bin/encryption-settings create \
  --cipher-algorithm AES \
  --key-length-bits 256 \
  --set-preferred
```

4. Obtain a list of the definitions in the encryption-settings database.

```
$ bin/encryption-settings list
```

5. You can export an encryption-settings definition from the database using a command like the following where the encryption-settings ID should be changed as necessary to suit your deployment:

```
$ bin/encryption-settings export \
  --id DA39A3EE5E6B4B0D3255BFEF95601890AFD80709 \
  --output-file /tmp/exported-key \
  --pin-file /tmp/exported-key.pin
```

6. If no PIN file is specified, then you will be interactively prompted to provide it. To import an encryption-settings definition into the database on another server.

```
$ bin/encryption-settings import \
  --input-file /tmp/exported-key \
  --pin-file /tmp/exported-key.pin \
  --set-preferred
```

Encrypting passphrase files

Using an encrypted passphrase or tools.properties file enables you to have credentials available for use by the server and command-line tools, but not stored in the clear. These files should be encrypted with the encrypt-file tool, including:

- Certificate keystore and truststore PIN files. When setting up an instance with encryption enabled and at least one of SSL or StartTLS enabled, the installer will automatically encrypt the PIN files for the config/keystore, config/truststore, and config/ads-truststore certificate databases.
- Passphrase files specified by command-line arguments. For example, most LDAP tools offer **`--bindPasswordFile`**, **`--keystorePasswordFile`**, and **`--truststorePasswordFile`** arguments.
- The **`config/tools.properties`** file, which will automatically be used to obtain a default set of arguments for most command-line tools. Alternately, the **`--propertiesFilePath`** argument can be used to specify an alternate properties file.

These files should be encrypted with the following considerations:

- If the file is encrypted with a key obtained from the server's encryption settings database, then the server and associated command-line tools should be able to retrieve the appropriate key from the encryption settings database so that the clear-text contents of the file can be accessed without any interaction (although if the cipher stream provider configured to protect the contents of the encryption

settings database requires interaction, like the wait for passphrase cipher stream provider, then command-line tools may require interaction to unlock the encryption settings database).

- If the file is encrypted with a passphrase that the user specifies rather than one obtained from the encryption settings database, then the user will be interactively prompted for that passphrase when running the tool. This option cannot be used for keystore and truststore PIN files that need to be accessed by the server.

To encrypt a file with the server's preferred encryption settings definition, use a command such as:

```
$ bin/encrypt-file --input-file password.txt \
  --output-file password.txt.encrypted
```

To use a key from an encryption settings definition that isn't the default, use the **`--encryption-settings-id`** argument to specify the ID of the desired encryption settings definition (which can be obtained with **`cencryption-settings list`**), such as:

```
$ bin/encrypt-file --input-file password.txt \
  --output-file password.txt.encrypted \
  --encryption-settings-id 4B6899D6716FC3AFFD71F7B447EB135063A0E724
```

To encrypt the file with a passphrase rather than a key from an encryption settings definition, use either the **`--prompt-for-passphrase`** argument to interactively prompt for the passphrase, or use the **`--passphrase-file`** argument to specify the path to a file containing the clear-text passphrase. For example:

```
$ bin/encrypt-file --input-file password.txt \
  --output-file password.txt.encrypted \
  --prompt-for-passphrase
```

About backing up and restoring the encryption-settings definitions

If using data encryption in a Directory Server instance, do not lose the encryption-settings definitions used to encrypt data in the server. If an encryption-settings definition is lost, any data encrypted with that definition will be completely inaccessible. Make sure the encryption-settings definitions are backed up regularly.

The Directory Server provides two different mechanisms for backing up and restoring encryption-settings definitions. One or more encryption-settings definitions can be exported and imported using the **`encryption-settings`** tool. Or, the entire encryption-settings database can be backed up and restored using the Directory Server's **`backup`** and **`restore`** tools.

If a pin file is used to define a passphrase to the encryption-settings database, the passphrase must be backed up and kept secure independently of the userRoot and encryption-settings database backups. The passphrase in the pin file is needed if the encryption-settings database is to be restored into a different server root.

Exporting encryption-settings definitions

About this task

To back up an individual definition (or to export it from one server so that you can import it into another), use the `export` subcommand to the **`encryption-settings`** command. The subcommand takes the following arguments:

- **--id {id}**. Specifies the ID for the encryption-settings definition to be exported. This argument can be specified multiple times. If it is omitted, all definitions are exported.
- **--output-file {path}**. Specifies the path to the output file to which the encryption-settings definition will be written. This argument is required.

- **--pin-file {path}**. Specifies the path to a PIN file containing the password to use to encrypt the contents of the exported definition. If this argument is not provided, then the PIN will be interactively requested from the server.

Steps

- Use the **`encryption-settings`** tool with the `export` subcommand to export the definition to a file.

```
$ bin/encryption-settings export --id
 F635E109A8549651025D01D9A6A90F7C9017C66D \
  --output-file /tmp/exported-key
Enter the PIN to use to encrypt the definition:
Re-enter the encryption PIN:
```

```
Successfully exported encryption settings definition
F635E109A8549651025D01D9A6A90F7C9017C66D to file /tmp/exported-key
```

Importing encryption-settings definitions

About this task

To import an encryption-settings definition that has been previously exported, use the **`encryption-settings`** tool with the `import` subcommand. The subcommand takes the following arguments:

- **--input-file {path}**. Specifies the path to the file containing the exported encryption-settings definition. This argument is required.
- **--pin-file {path}**. Specifies the path to a PIN file containing the password to use to encrypt the contents of the exported definition. If this argument is not provided, then the PIN will be interactively requested from the server.
- **--set-preferred**. Specifies that the newly-imported encryption-settings definition should be made for the preferred definition for subsequent encryption-settings.

Steps

- Use the **`encryption-settings`** tool with the `import` subcommand to import the definition to a file.

```
$ bin/encryption-settings import --input-file /tmp/exported-key --set-
preferred
Enter the PIN used to encrypt the definition:
```

```
Successfully imported encryption settings definition
F635E109A8549651025D01D9A6A90F7C9017C66D from file /tmp/exported-key
```

Enabling data encryption in the server

About this task

Encryption can be enabled during server setup, by defining an encryption key and passphrase. This configuration should be used across all servers in a topology. On legacy systems or post setup, data encryption can be configured by having at least one encryption-settings definition available for use. Then, set the value of the `encrypt-data` global configuration property to true.

Setting the global configuration property will automatically enable data encryption for all types of backends that support it (including the changelog backend and indexes), as well as for the replication server database. All subsequent write operations will cause the corresponding records written into any of these locations to be encrypted. Any existing data will remain unencrypted until it is rewritten by a write operation. If you wish to have existing data encrypted, then you will need to export that data to LDIF and re-import it. This will work for both the data backends, the changelog, and indexes, but it is not an option for the replication database, so existing change records will remain unencrypted until they are purged. If this

is not considered acceptable in your environment, then follow the steps in *Dealing with a Compromised Encryption Key* to safely purge the replication database.

Configuration for backups and LDIF exports can be done with the following global properties:

- **automatically-compress-encrypted-ldif-exports**. Indicates whether to automatically compress LDIF exports that are also encrypted. If set to true, any LDIF export that is encrypted (either explicitly with **--encryptLDIF**,or implicitly with the **encrypt-ldif-exports-by-default** configuration property) will automatically be gzip-compressed. If this is false, encrypted LDIF exports can still be manually compressed using the **--compress** command-line argument.
- **backup-encryption-settings-definition-id**. The unique identifier for the encryption settings definition to use to generate the encryption key for encrypted backups by default. If this property is given a value, then a definition with that ID must exist in the server's encryption settings database. If this property is not given a value, but the server is configured with at least one encryption settings definition, then the preferred definition is used. If no encryption settings definitions are available, the server will use an internal key shared among servers in the topology. Regardless of this property's value, it can be overridden with the **backup** command-line tool. Providing one of the **--promptForEncryptionPassphrase** or **--encryptionPassphraseFile** arguments will generate the encryption key from the provided passphrase. Or, the **--encryptionSettingsDefinitionID** argument can be used to generate the key from the specified encryption settings definition.
- **encrypt-backups-by-default**. Indicates whether the server should encrypt backups by default. If set to true, a defined **backup-encryption-settings-definition-id** value will be used to generate the encryption key for the backup. If this property is true, and if a **backup-encryption-settings-definition-id** value is not specified, the server will try to use the preferred encryption settings definition to generate the encryption key. If the server is not configured with any encryption settings definitions, an internal key that is shared among instances in the topology is used. Regardless of this property's value, it can be overridden with the **backup** command-line tool's **--encrypt** argument, even if this property is set to false. The **--doNotEncrypt** argument will always cause the backup to be unencrypted, even if this property has a value of true.
- **encrypt-ldif-exports-by-default**. Indicates whether the server should encrypt LDIF exports by default. If set to true, and an **ldif-export-encryption-settings-definition-id** value is specified, then that encryption settings definition is used to generate the encryption key for the export. If this property is true, and an **ldif-export-encryption-settings-definition-id** value is not specified, the server will first try to use the preferred encryption settings definition to generate the encryption key. If the server is not configured with any encryption settings definitions, then an internal key that is shared among instances in the topology is used. Regardless of this property's value, the default behavior can be overridden with the **export-ldif** command-line tool. The tool's **--encryptLDIF** argument will always encrypt the export, and the **--doNotEncryptLDIF** argument will always create an unencrypted export.

Steps

- Use **dsconfig** to set the global configuration property for data encryption to true.

```
$ bin/dsconfig set-global-configuration-prop --set encrypt-data:true
```

Using data encryption in a replicated environment

Data encryption is only used for the on-disk storage for data within the server. Whenever clients access that data, it is presented in unencrypted form (although the communication with those clients may itself be encrypted using SSL or StartTLS). Replication, the communication of updates between replication servers, is always encrypted using SSL. Each server may apply data encryption in a completely independent manner and have different sets of encryption-settings definitions. It is also possible to have a replication topology containing some servers with data encryption enabled and others with it disabled.

However, when initializing the backend of one server from another server with data encryption enabled, then the server being initialized must have access to all encryption-settings definitions that may have been used for data contained in that backend. To do this, perform an export of the encryption-settings database

on the source server using **`bin/encryption-settings export`** and import it on the target server using **`bin/encryption-settings import`**.

Dealing with a compromised encryption key

About this task

If an encryption-settings definition becomes compromised such that an unauthorized individual obtains access to the encryption key, then any data encrypted with that definition is also vulnerable because it can be decrypted using that key. It is very important that the encryption-settings database be protected (for example, using file permissions or file system ACLs) to ensure that its contents remain secure.

In the event that an encryption-settings definition is compromised, stop using the definition immediately. Any data encrypted with the compromised key should be re-encrypted with a new definition or purged from the server. This can be done on one server at a time to avoid an environment-wide downtime, but it should be completed as quickly as possible on all servers that had used that definition at any point in the past in order to minimize the risk of that data becoming exposed.

The recommended process for responding to a compromised encryption settings definition is as follows:

Steps

1. Create a new encryption-settings definition and make it the preferred definition for new writes.
2. Ensure that client traffic is routed away from the server instance to be updated. For example, if the Directory Server is accessed through a Directory Proxy Server, then you may set the `health-check-state` configuration property for any LDAP external server definitions that reference that server to have a value of unavailable.
3. Ensure that external clients are not allowed to write operations in the directory server instance. This may be accomplished by setting the `writability-mode` global configuration property to have a value of `internal-only`.
4. Wait for all outstanding local changes to be replicated out to other servers. This can be accomplished by looking at the monitor entries with the `ds-replication-server-handler-monitor-entry` object class to ensure that the value of the `update-sent` attribute is no longer increasing.
5. Stop the directory server instance.
6. Delete the replication server database by removing all files in the `changeLogDb` directory below the server root. As long as all local changes have been replicated out to other servers, this will not result in any data loss in the replication environment.
7. Export the contents of all local DB and changelog backends to LDIF. Then, re-import the data from LDIF, which will cause it to be encrypted using the new preferred encryption settings definition.
8. Export the compromised key from the encryption settings database to back it up in case it may be needed again in the future (e.g., if some remaining data was later found to have been encrypted with the key contained in that definition). Then, delete it from the encryption settings database so that it can no longer be used by that directory server instance.
9. Start the directory server instance.
10. Allow replication to bring the server back up-to-date with any changes processed while it was offline.
11. Re-allow externally-initiated write operations by changing the value of the global `writability-mode` configuration property back to enabled.
12. Re-configure the environment to allow client traffic to again be routed to that server instance (e.g., by changing the value of the "health-check-state" property in the corresponding LDAP external instance definitions in the Directory Proxy Server instances back to "dynamically-determined").

Configuring sensitive attributes

Data encryption is only applied to the on-disk storage for a Directory Server instance. It does not automatically protect information accessed or replicated between servers, although the server offers other

mechanisms to provide that protection (i.e., SSL, StartTLS, SASL). Ensuring that all client communication uses either SSL or StartTLS encryption and ensuring that all replication traffic uses SSL encryption ensures that the data is protected from unauthorized individuals who may be able to eavesdrop on network communication. This communication security may be enabled independently of data encryption (although if data encryption is enabled, then it is strongly recommended that secure communication be used to protect network access to that data).

However, for client data access, it may not be as simple as merely enabling secure communication. In some cases, it may be desirable to allow insecure access to some data. In other cases, it may be useful to have additional levels of protection in place to ensure that some attributes are even more carefully protected. These kinds of protection may be achieved using sensitive attribute definitions.

Each sensitive attribute definition contains a number of configuration properties, including:

- **`attribute-type`** . Specifies the set of attribute types whose values may be considered sensitive. At least one attribute type must be provided, and all specified attribute types must be defined in the server schema.
- **`include-default-sensitive-operational-attributes`** . Indicates whether the set of sensitive attributes should automatically be updated to include any operational attributes maintained by the Directory Server itself that may contain sensitive information. At present, this includes the `ds-sync-hist` operation attribute, which is used for data required for replication conflict resolution and may contain values from other attributes in the entry.
- **`allow-in-filter`** . Indicates whether sensitive attributes may be used in filters. This applies not only to the filter used in search requests, but also filters that may be used in other places, like the assertion and join request controls. The value of this property must be one of:
  - `Allow` (allow sensitive attributes to be used in filters over both secure and insecure connections)
  - `Reject` (reject any request which includes a filter targeting one or more sensitive attributes over both secure and insecure connections)
  - `Secure-only` (allow sensitive attributes to be used in filters over secure connections, but reject any such requests over insecure connections)
- **`allow-in-add`** . Indicates whether sensitive attributes may be included in entries created by LDAP add operations. The value of this property must be one of:
  - `Allow` (allow sensitive attributes to be included in add requests over both secure and insecure connections)
  - `Reject` (reject any add request containing sensitive attributes over both secure and insecure connections)
  - `Secure-only` (allow sensitive attributes to be included in add requests received over a secure connection, but reject any such requests over an insecure connection)
- **`allow-in-compare`** . Indicates whether sensitive attributes may be targeted by the assertion used in a compare operation. The value of this property must be one of:
  - `Allow` (allow sensitive attributes to be targeted by requests over both secure and insecure connections)
  - `Reject` (reject any compare request targeting a sensitive attribute over both secure and insecure connections)
  - `Secure-only` (allow compare requests targeting sensitive attributes over a secure connection, but reject any such requests over an insecure connection)
- **`allow-in-modify`** . Indicates whether sensitive attributes may be updated using modify operations. The value of this property must be one of:
  - `Allow` (allow sensitive attributes to be modified by requests over both secure and insecure connections)
  - `Reject` (reject any modify request updating a sensitive attribute over both secure and insecure connections)
  - `Secure-only` (only modify requests updating sensitive attributes over a secure connection, but reject any such request over an insecure connection)

The `allow-in-returned-entries`, `allow-in-filter`, `allow-in-add`, `allow-in-compare`, and `allow-in-modify` properties all have default values of `secure-only`, which prevents the possibility of exposing sensitive data in the clear to anyone able to observe network communication.

If a client connection policy references a sensitive attribute definition, then any restrictions imposed by that definition will be enforced for any clients associated with that client connection policy. If multiple sensitive attribute definitions are associated with a client connection policy, then the server will use the most restrictive combination of all of those sets.

Note that sensitive attribute definitions work in conjunction with other security mechanisms defined in the server and may only be used to enforce additional restrictions on clients. Sensitive attribute definitions may never be used to grant a client additional access to information that it would not have already had through other means. For example, if the `employeeSSN` attribute is declared to be a sensitive attribute and the `allow-in-returned-entries` property has a value of `Secure-only`, then the `employeeSSN` attribute will only be returned to those clients that have both been granted permission by the access control rules defined in the server and are communicating with the server over a secure connection. The `employeeSSN` attribute will be stripped out of entries returned to clients normally authorized to see it if they are using insecure connections, and it will also be stripped out of entries for clients normally not authorized to see it even if they have established secure connections.

Creating a sensitive attribute

Steps

1. To create a sensitive attribute, you must first create one or more sensitive attribute definitions.

   For example, to create a sensitive attribute definition that will only allow access to the `employeeSSN` attribute by clients using secure connections, the following configuration changes may be made:

   ```
   $ bin/dsconfig create-sensitive-attribute \
     --attribute-name "Employee Social Security Numbers" \
     --set attribute-type:employeeSSN \
     --set include-default-sensitive-operational-attributes:true \
     --set allow-in-returned-entries:secure-only \
     --set allow-in-filter:secure-only \
     --set allow-in-add:secure-only \
     --set allow-in-compare:secure-only \
     --set allow-in-modify:secure-only
   ```

2. Associate those sensitive attribute definitions with the client connection policies for which you want them to be enforced.

   ```
   $ bin/dsconfig set-client-connection-policy-prop --policy-name default \
     --set "sensitive-attribute:Employee Social Security Numbers"
   ```

Configuring global sensitive attributes

About this task

Administrators can assign one or more sensitive attribute definitions to a client connection policy. However, in an environment with multiple client connection policies, it could be easy to add a sensitive attribute definition to one policy but overlook it in another. The Directory Server supports the ability to define sensitive attributes as a global configuration option so that they will automatically be used across all client connection policies.

Steps

- Run **dsconfig** to add a global sensitive attribute across all client connection policies. The following command adds the `employeeSSN` as a global sensitive attribute, which is applied across all client connection policies.

```
$ bin/dsconfig set-global-configuration-prop --add "sensitive-
attribute:employeeSSN"
```

Excluding a global sensitive attribute on a client connection policy

About this task

Administrators can set a global sensitive attribute across all client connection policies. However, there may be cases when a specific directory server must exclude the sensitive attribute as it may not be needed for client connection requests. For example, in most environments it is good to declare the `userPassword` attribute to be a sensitive attribute in a manner that prevents it from being read by external clients. Further, this solution is more secure than protecting the `password` attribute using the server's default global ACI, which only exists for backwards compatibility purposes. If the Data Sync Server is installed, then it does need to be able to access passwords for synchronization purposes. In this case, the administrator can set `userPassword` to be a sensitive attribute in all client connection policies, but exclude it in a policy specifically created for use by the Data Sync Server. The Directory Server provides an `exclude-global-sensitive-attribute` property for this purpose.

Steps

1. Run **dsconfig** to remove the global ACI that limits access to the `userPassword` or `authPassword` attribute. This is present for backwards compatibility.

```
$ bin/dsconfig set-access-control-handler-prop \
  --remove 'global-aci:(targetattr="userPassword || authPassword")
  (version 3.0; acl "Prevent clients from retrieving passwords from the
 server";
  deny (read,search,compare) userdn="ldap:///anyone";)'
```

2. Run **dsconfig** to add the `userPassword` attribute as a global sensitive attribute, which is applied to all client connection policies. Do this by adding the built-in "Sensitive Password Attributes" Sensitive Attribute definition to the Global Configuration.

```
$ bin/dsconfig set-global-configuration-prop \
  --add "sensitive-attribute:Sensitive Password Attributes"
```

3. If the server is designated to synchronize passwords with a Sync Server, then it is necessary to configure a client connection policy for the Sync User to exclude the global sensitive attribute. The following is an example on how to create a new policy if the Data Sync Server binds with the default DN of `cn=Sync User,cn=Root DNs,cn=config`.

```
$ bin/dsconfig create-connection-criteria \
  --criteria-name "Requests by Sync Users" \
  --type simple \
  --set user-auth-type:internal \
  --set user-auth-type:sasl \
  --set user-auth-type:simple \
  --set "included-user-base-dn:cn=Sync User,cn=Root DNs,cn=config"

$ bin/dsconfig create-client-connection-policy \
  --policy-name "Data Sync Server Connection Policy" \
  --set enabled:true \
  --set evaluation-order-index:9998 \
  --set "connection-criteria:Requests by Sync Users" \
  --set "exclude-global-sensitive-attribute:Sensitive Password Attributes"
```

Working with the LDAP Changelog

The Directory Server provides a client-accessible LDAP changelog (based on the Changelog Internet Draft Specification) for the purpose of allowing other LDAP clients to retrieve changes made to the server in standard LDAP format. The LDAP changelog is typically used by external software to maintain application compatibility between client services.

Overview of the LDAP changelog

The Directory Server provides a client-accessible LDAP changelog (based on the Changelog Internet Draft Specification) for the purpose of allowing other LDAP clients to retrieve changes made to the server in standard LDAP format. The LDAP changelog is typically used by external software to maintain application compatibility between client services. For example, you can install the Data Sync Server that monitors the LDAP changelog for any updates that occur on a source directory server and synchronizes these changes to a target DIT or database server. The Directory Server provides an additional feature in that the LDAP changelog supports virtual attributes.

ⓘ **Note:** The LDAP Changelog should not be confused with the Replication Changelog. The main distinction is as follows:

- The LDAP Changelog (i.e., the external changelog that clients can access) physically resides at `<server-root>/db/changelog`.
- The Replication Changelog Backend (i.e., the changelog that replication servers use) physically resides at `<server-root>/changelogDB`.

Key changelog features

As of version 3.2, the Directory Server supports two new Changelog Backend properties that allow access control filtering and sensitive attribute evaluation for targeted entries. External client applications can change the contents of attributes they can see in the targeted entry based on the access control rules applied to the associated base DN.

- **apply-access-controls-to-changelog-entry-contents**. Indicates whether the contents of changelog entry attributes (i.e., `changes`, `deletedEntryAttrs`, `ds-changelog-entry-key-attr-values`, `ds-changelog-before-values`, and `ds-changelog-after-values`) are subject to access control and/or sensitive attribute evaluation to limit data that LDAP clients can see. The client must have the access control permissions to read changelog entries to retrieve them in any form. If this feature is enabled and the client does not have permission to read an entry at all, or if that client does not have permission to see any attributes that were targeted by the change, then the associated changelog entries targeted by those operations will be suppressed. If a client does not have permission to see certain attributes within the target entry, then references to those attributes in the changelog entry will also be suppressed. This property only applies to standard LDAP searches of the `cn=changelog` branch.
- **report-excluded-changelog-attributes**. Indicates whether to include additional information about any attributes that may have been removed due to access control filtering. This property only applies to content removed as a result of processing performed by the `apply-access-controls-to-changelog-entry-contents` property. Possible values are:
  - **none** - Indicates that changelog entries should not include any information about attributes that have been removed.
  - **attribute-counts** - Indicates that changelog entries should include a count of user and/ or operational attributes that have been removed. If any user attribute information was excluded from a changelog entry, the number of the excluded user attributes will be reported in the `ds-changelog-num-excluded-user-attributes` attribute of the changelog entry. If any operational attribute information was excluded from a changelog entry, then the number of the excluded operational attributes will be reported in the `ds-changelog-num-excluded-operational-attributes`

attribute of the changelog entry. Both the `ds-changelog-num-excluded-user-attributes` and `ds-changelog-num-excluded-operational-attributes` are operational and must be explicitly requested by clients (or all operational attributes requested using "+") to be returned.

- **attribute-names** - Indicates that changelog entries should include the names of user and/or operational attributes that have been removed. If any user attribute information was excluded from a changelog entry, then the names of the excluded user attributes will be reported in the `ds-changelog-excluded-user-attributes` attribute of the changelog entry. If any operational attribute information was excluded from a changelog entry, then the names of the excluded operational attributes will be reported in the `ds-change-log-excluded-operational-attribute` attribute of the changelog entry. Both the `ds- changelog-excluded-user-attribute` and `ds-changelog-excluded-operational-attribute` attributes are operational and must be explicitly requested by clients (or all operational attributes requested via "+") to be returned.

Enabling access control filtering in the LDAP changelog

About this task

To set up access control to the LDAP Changelog, use the **`dsconfig`** tool to enable the properties to the Changelog Backend. Only admin users with the `bypass-acl` privilege can read the changelog.

Steps

1. Enable the `apply-access-control-to-changelog-entry-contents` property to allow LDAP clients to undergo access control filtering using standard LDAP searches of the `cn=changelog` backend.

   ```
   $ bin/dsconfig set-backend-prop --backend-name "changelog" \
     --set "apply-access-controls-to-changelog-entry-contents:true"
   ```

   Access control filtering will be applied regardless of the value of the `apply-access-controls-to-changelog-entry-contents` setting when the Changelog Backend is servicing requests from an PingDataSync Server that has the `filter-changes-by-user` Sync Pipe property set.
2. Optional. Set the `report-excluded-changelog-attributes` property to include a count of users that have been removed through access control filtering. The count appears in the `ds-changelog-num-excluded-user-attributes` attribute for users and the `ds-changelog-num-excluded-operational-attributes` attribute for operational attributes.

   ```
   $ bin/dsconfig set-backend-prop --backend-name "changelog" \
     --set "report-excluded-changelog-attributes:attribute-counts"
   ```

Useful changelog features

The Directory Server provides two useful changelog configuration properties: `changelog-max-before-after-values` and `changelog-include-key-attribute`.

- **changelog-max-before-after-values**. Setting this property to a non-zero value causes all of the old values and all of the new values (up to the specified maximum) for each changed attribute to be stored in the changelog entry. The values will be stored in the `ds-change-log-before-values` and `ds-`

`changelog-after-values` attributes on the changelog entry. These attributes are not present by default.

> ⓘ **Note:** The `changelog-max-before-after-values` property can be expensive for attributes with hundreds or thousands of values, such as a group entry.

If any attribute has more than the maximum number of values, their names and number of before/after values will be stored in the `ds-changelog-attr-exceeded-max-values-count` attribute on the changelog entry. This is a multi-valued attribute whose format is:

```
attr=attributeName,beforeCount=100,afterCount=101
```

where "attributeName" is the name of the attribute and the "beforeCount" and "afterCount" are the total number of values for that attribute before and after the change, respectively. This attribute indicates that you need to reset the `changelog-max-before-after-values` property to a higher value. When this attribute is set, an alert will be generated.

> ⓘ **Note:** If the number of values for an attribute exceeds the maximum value set by the `changelog-max-before-after-values` property, then those values will not be stored.

- **changelog-include-key-attribute**. This property is used for correlation attributes that need to be synchronized across servers, such as `uid`. It causes the current (after-change) value of the specified attributes to be recorded in the `ds-changelog-entry-key-attr-values` attribute on the changelog entry. This applies for all change types. On a DELETE operation, the values are from the entry before they were deleted.

  The key values will be recorded on every change and override the settings configured in `changelog-include-attribute`, `changelog-exclude-attribute`, `changelog-deleted-entry-include-attribute`, or `changelog-deleted-entry-exclude-attribute`.

Example of the changelog features

After the `changelog-max-before-after-values` property is set, the before-and-after values of any change attribute will be recorded in the LDAP Changelog. For example, given a simple entry with two multi-valued mail attributes:

```
dn: uid=test,dc=example,dc=com
objectclass: inetorgperson
cn: test user
sn: user
description: oldDescription
mail: test@yahoo.com
mail: test@gmail.com
```

Then, apply the following changes to the entry:

```
dn: uid=test,dc=example,dc=com
changetype: modify
add: mail
mail: test@hotmail.com
-
delete: mail
mail: test@yahoo.com
-
replace: description
description: newDescription
```

The resulting changelog would record the following attribute values:

```
dn: changeNumber=1,cn=changelog
```

```
objectClass: top
objectClass: changeLogEntry
targetDN: uid=test,dc=example,dc=com
changeType: modify
changes::
YWRkOiBtYWlsCm1haWw6IHRlc3RAaG90bWFpbC5jb20KLQpkZWxldGU6IG1haWwKbWFpbDogdGVzdEB5YWh
vby5jb20KLQpyZXBsYWNlOiBkZXNjcmlwdGlvbgpkZXNjcmlwdGlvbjogbmV3RGVzY3JpcHRpb24KLQpyZX
BsYWNlOiBtb2RpZmllcnNOYW1lCm1vZGlmaWVyc05hbWU6IGNuPURpcmVjdG9yeSBNYW5hZ2VyLGNuPVJvb
3QgRE5zLGNuPWNvbmZpZwotCnJlcGxhY2U6IGRzLXVwZGF0ZS10aW1lCmRzLXVwZGF0ZS10aW1lOjogQUFB
QkxxQitIaTQ9Ci0KAA==
ds-changelog-before-values::
 ZGVzY3JpcHRpb246IG9sZERlc2NyaXB0aW9uCm1haWw6IHRlc3RAeW
Fob28uY29tCm1haWw6IHRlc3RAZ21haWwuY29tCmRzLXVwZGF0ZS10aW1lOjogQUFBQkxxQjdaZ1E9Cm1vZ
GlmaWVyc05hbWU6IGNuPURpcmVjdG 9yeSBNYW5hZ2VyLGNuPVJvb3QgRE5zLGNuPWNvbmZpZwo=
ds-changelog-after-values::
 ZGVzY3JpcHRpb246IG5ld0Rlc2NyaXB0aW9uCm1haWw6IHRlc3RAZ21
haWwuY29tCm1haWw6IHRlc3RAaG90bWFpbC5jb20KZHMtdXBkYXRlLXRpbWU6OiBBQUFCTHFCK0hpND0KbW
9kaWZpZXJzTmFtZTogY249RGlyZWN0b3J5IE1hbmFnZXIsY249Um9vdCBETnMsY249Y29uZmlnCg==
ds-changelog-entry-key-attr-values:: dWlkOiB0ZXN0Cg==
changenumber: 1
```

You can run the **bin/base64** `decode -d` command-line tool to view the decoded value for the changes, `ds-changelog-before-values`, `ds-changelog-after-values` attributes:

After base64 decoding, the `changes` attribute reads:

```
add: mail
mail: test@hotmail.com
-
delete: mail
mail: test@yahoo.com
-
replace: description
description: newDescription
-
replace: modifiersName
modifiersName: cn=Directory Manager,cn=Root DNs,cn=config
-
replace: modifyTimestamp
modifyTimestamp: 20131010020345.546Z
-
```

After base64 decoding, the `ds-changelog-before-values` attribute reads:

```
description: oldDescription
mail: test@yahoo.com
mail: test@gmail.com
modifyTimestamp: 20131010020345.546Z
modifiersName: cn=Directory Manager,cn=Root DNs,cn=config
```

After base64 decoding, the `ds-changelog-after-values` attribute reads:

```
description: newDescription
mail: test@gmail.com
mail: test@hotmail.com
modifyTimestamp: 20131010020345.546Z
modifiersName: cn=Directory Manager,cn=Root DNs,cn=config
```

Viewing the LDAP changelog properties

You can view the LDAP changelog properties by running the **dsconfig** `get-backend-prop` command and specifying the changelog backend.

Viewing the LDAP changelog properties using dsconfig non-interactive mode

Steps

- Use **dsconfig** to view the changelog properties on the Directory Server. To view "advanced" properties that are normally hidden, add the `--advanced` option when running the command. For a specific description of each property, see the *PingDirectory Server Configuration Reference*.

```
$ bin/dsconfig get-backend-prop --backend-name changelog
```

```
Property                                    : Value(s)
--------------------------------------------:-------------
backend-id                                  : changelog
description                                 : -
enabled                                     : false
writability-mode                            : disabled
base-dn                                     : cn=changelog
set-degraded-alert-when-disabled            : false
return-unavailable-when-disabled            : false
db-directory                                : db
db-directory-permissions                    : 700
changelog-maximum-age                       : 2 d
db-cache-percent                            : 1
changelog-include-attribute                 : -
changelog-exclude-attribute                 : -
changelog-deleted-entry-include-attribute   : -
changelog-deleted-entry-exclude-attribute   : -
changelog-include-key-attribute             : -
changelog-max-before-after-values           : 0
changelog-write-batch-size                  : 100
changelog-purge-batch-size                  : 1000
changelog-write-queue-capacity              : 100
write-lastmod-attributes                    : true
use-reversible-form                         : false
je-property                                 : -
```

Enabling the LDAP changelog

By default, the LDAP changelog is disabled on the Directory Server. If you are using the **dsconfig** tool in interactive mode, the changelog appears in the Backend configuration as a Standard object menu item.

ⓘ **Note:** You can enable the feature using the **dsconfig** tool only if required as it can significantly affect LDAP update performance.

Enabling the LDAP changelog using dsconfig non-interactive mode

Steps

- Use **dsconfig** to enable the changelog property on the Directory Server.

```
$ bin/dsconfig set-backend-prop \
  --backend-name changelog --set enabled:true
```

Enabling the LDAP changelog using interactive mode

Steps

1. Use **dsconfig** to enable the changelog on each server in the network. Then, authenticate to the server by entering the host name, LDAP connection, port, bindDN and bind password.

   ```
   $ bin/dsconfig
   ```
2. On the Directory Server main menu, type `o` to change from the Basic object level to the Standard object level.
3. Enter the option to select the Standard Object level.
4. On the Directory Server main menu, type the number corresponding to Backend.
5. On the **Backend Management** menu, enter the option to view and edit an existing backend.
6. Next, you should see a list of the accessible backends on your system. For example, you may see options for the `changelog` and `userRoot` backends. Enter the option to work with the changelog backend.
7. On the **Changelog Backend properties** menu, type the number corresponding to the Enabled property.
8. On the **Enabled Property** menu, type the number to change the `Enabled` property to TRUE.
9. On the **Backend Properties** menu, type `f` to apply the change. If you set up the server in a server group, type `g` to update all of the servers in the group. Otherwise, repeat steps 1-10 on the other servers.
10. Verify that changes made to the data are recorded in the changelog.

Changing the LDAP changelog database location

In cases where disk space issues arise, you can change the on-disk location of the LDAP Change Log database. The changelog backend supports a `db-directory` property that specifies the absolute or relative path (relative to the local server root) to the file system directory that is used to hold the Oracle Berkeley DB Java Edition database files containing the data for this backend.

If you change the changelog database location, you must stop and then restart the Directory Server for the change to take effect. If the changelog backend is already enabled, then the database files must be manually moved or copied to the new location while the server is stopped.

Changing the LDAP changelog location using dsconfig non-interactive mode

Steps

1. Use **dsconfig** to change the database location for the LDAP Changelog, which by default is at `<server-root>/db`. The following command sets the LDAP changelog backend to `<server-root>/db2`. Remember to include the LDAP connection parameters (for example, host name, port, bindDN, bindPassword).

   ```
   $ bin/dsconfig set-backend-prop --backend-name changelog \
     --set "db-directory:db2" --set "enabled:true"
   ```

   The database files are stored under `<server-root>/db2/changelog`. The files for this backend are stored in a sub-directory named after the `backend-id` property.
2. Stop and restart the server. Since the LDAP changelog backend was previous disabled, there is no need to manually relocate any existing database files.

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

Resetting the LDAP changelog location using dsconfig non-interactive mode

Steps

1. If you have changed the LDAP Changelog location, but want to reset it to its original location, use **`dsconfig`** to reset it. The following command resets the LDAP changelog backend to `<server-root>/db` location. Remember to include the LDAP connection parameters (for example, host name, port, bindDN, bindPassword).

   ```
   $ bin/dsconfig set-backend-prop --backend-name changelog \
     --reset "db-directory"
   ```

2. The server attempts to use whatever it finds in the configured location when it starts. If there is nothing there, it will create an empty database. If the LDAP changelog backend at the previous location is enabled at the time, stop the server, manually copy the database files to the new LDAP changelog location, and then restart the server.

Viewing the LDAP changelog parameters in the Root DSE

About this task

The Root DSE is a special entry that holds operational information about the server. The entry provides information about the LDAP controls, extended operations, and SASL mechanisms available in the server as well as the state of the data within the changelog. For changelog parameters, the attributes of interest include:

- **firstChangeNumber**. Change number for the first (oldest) change record contained in the LDAP changelog.
- **lastChangeNumber**. Change number for the last (most recent) change record contained in the LDAP changelog.
- **lastPurgedChangeNumber**. Change number for the last change that was purged from the LDAP changelog. It can be 0 if no changes have yet been purged.
- **firstReplicaChange**. Information about the first (oldest) change record for a change received from the specified replica. This is a multi-valued attribute and should include a value for each server in the replication topology.
- **lastReplicaChange**. Information about the last (most recent) change record for a change received from the specified replica.

The `firstReplicaChange` and `lastReplicaChange` attributes use the following syntax:

```
serverID:CSN:changeNumber
```

where:

- **serverID**. Specifies the unique identifier for the server updating the change log.
- **CSN**. Specifies the Change Sequence Number, which is the time when the update was made to the given replica.
- **changeNumber**. Specifies the order of the change that is logged to the LDAP changelog.

The `firstReplicaChange` and `lastReplicaChange` attributes can be used to correlate information in the local LDAP Change Log with data in the LDAP Change Log of other servers in the replication topology. The order of the individual changes in the LDAP Change Log can vary between servers based on the order in which they were received from a replica.

Steps

- Use **`ldapsearch`** to view the Root DSE.

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

Viewing the LDAP changelog using ldapsearch

All records in the changelog are immediate children of the `cn=changelog` entry and are named with the `changeNumber` attribute. You can view changelog entries using **`ldapsearch`**. Changes are represented in the form documented in the *draft-good-ldap-changelog* specification with the targetDN attribute providing the DN of the updated entry, the `changeType` attribute providing the type of operation (add, delete, modify, or modDN), and the changes attribute providing a base64-encoded representation of the attributes included in the entry (for add operations) or the changes made (for modify operations) in LDIF form. You can view the changes by decoding the encoded value using the **`base64`** `decode` utility. The LDAP SDK for Java also provides support for parsing changelog entries.

Viewing the LDAP changelog using ldapsearch

Steps

1. Grant access to the `cn=changelog` backend to the `uid=admin` account using access control rules. By default, only the root user has access to this backend.

```
$ bin/ldapmodify
dn: cn=changelog
changetype: modify
add: aci
aci: (targetattr="*||+")
  (version 3.0; acl "Access to the changelog backend for the admin
 account";
  allow (all) userdn="ldap:///uid=admin,dc=example,dc=com";)
```

2. Use **`ldapsearch`** to view the changelog.

```
$ bin/ldapsearch --baseDN cn=changelog --dontWrap "(objectclass=*)"
```

```
dn: cn=changelog
objectClass: top
objectClass: untypedObject
cn: changelog

dn: changeNumber=1,cn=changelog
objectClass: changeLogEntry
objectClass: top
targetDN: uid=user.0,ou=People,dc=example,dc=com
changeType: modify
changes::
 cmVwbGFjZTogbW9iaWxlCm1vYmlsZTogKzEgMDIwIDE1NCA5Mzk4Ci0KcmVwbGFjZToga
G9tZVBob25lCmhvbWVQaG9uZTogKzEgMjI1IDIxNiA0OTQ5Ci0KcmVwbGFjZTogZ2l2ZW5OYW1lCmdp
dmVuTmFtZTogQWFyb24KLQpyZXBsYWNlOiBkZXNjcmlwdGlvbgpkZXNjcmlwdGlvbjogdGhpcyBpcyB
0aGUgZGVzY3JpcHRpb24gZm9yIEFhcm9uIEF0cC4KLQpyZXBsYWNlOiBtb2RpZmllcnNOYW1lCm1vZG
lmaWVyc05hbWU6IGNuPURpcmVjdG9yeSBNYW5hZ2VyLGNuPVJvb3QgRE5zLGNuPWNvbmZpZwotCnJlc
GxhY2U6IGRzLXVwZGF0ZS10aW1lCmRzLXVwZGF0ZS10aW1lOjogQUFBQkhQOHpUR0E9Cgo=
changenumber: 1
dn: changeNumber=2,cn=changelog
objectClass: changeLogEntry
objectClass: top
targetDN: dc=example,dc=com
changeType: modify
changes::
 cmVwbGFjZTogZHMtc3luYy1zdGF0ZQpkcy1zeW5jLXN0YXRlOiAwMDAwMDExQ0ZGMzM0Q
zYwNDA5MzAwMDAwMDAyCgo=
changenumber: 2
```

Viewing the LDAP change sequence numbers

Steps

- The changelog displays the server state information, which is important for failover between servers during synchronization operations. The server state information is exchanged between the servers in the network (LDAP servers and replication servers) as part of the protocol start message. It also helps the client application determine which server is most up-to-date. Make sure that the `uid=admin` account has the necessary access rights to the `cn=changelog` backend.

```
$ bin/ldapsearch --baseDN cn=changelog --dontWrap "(objectclass=*)" "+"
```

```
dn: cn=changelog

dn: changeNumber=1,cn=changelog
entry-size-bytes: 182
targetUniqueId: 68147342-1f61-3465-8489-3de58c532130
changeTime: 20111023002624Z
lastReplicaCSN: 0000011D27184D9E303000000001
replicationCSN: 0000011D27184D9E303000000001
replicaIdentifier: 12336

dn: changeNumber=2,cn=changelog
entry-size-bytes: 263
targetUniqueId: 4e9b7847-edcb-3791-b11b-7505f4a55af4
changeTime: 20111023002624Z
lastReplicaCSN: 0000011D27184F2E303000000002
replicationCSN: 0000011D27184F2E303000000002
replicaIdentifier: 12336
```

Viewing LDAP changelog monitoring information

About this task

The changelog contains a monitor entry that you can access over LDAP, JConsole, the Administrative Console, or SNMP. Make sure that the `uid=admin` account has the necessary access rights to the `cn=changelog` backend.

Steps

- Use **`ldapsearch`** to view the changelog monitor entry.

```
$ bin/ldapsearch --baseDN cn=changelog,cn=monitor "(objectclass=*)"
```

```
dn: cn=changelog,cn=monitor
objectClass: top
objectClass: ds-monitor-entry
objectClass: extensibleObject
cn: changelog
changelog: cn=changelog
firstchangenumber: 1
lastchangenumber: 8
lastpurgedchangenumber: 0
firstReplicaChange: 16225:0000011D0205237F3F6100000001:5
firstReplicaChange: 16531:0000011CFF334C60409300000002:1
lastReplicaChange: 16225:0000011D02054E8B3F6100000002:7
lastReplicaChange: 16531:0000011CFF334C60409300000002:1
oldest-change-time: 20081015063104Z
...(more data)...
```

Indexing the LDAP changelog

The Directory Server supports attribute indexing in the Changelog Backend to allow Get Changelog Batch requests to filter results that include only changes involving specific attributes. Normally, the directory server that receives a request must iterate over the whole range of changelog entries and then match entries based on search criteria for inclusion in the batch. The majority of this processing also involves determining whether the changelog entry includes changes to a particular attribute or set of attributes, or not. Using changelog indexing, client applications can dramatically speed up throughput when targeting the specific attributes.

Administrators can configure attribute indexing using the `index-include-attribute` and `index-exclude-attribute` properties on the Changelog Backend. The properties can accept the specific attribute name or special LDAP values "*" to specify all user attributes or "+" to specify all operational attributes.

To determine if the directory server supports this feature, administrators can view the Root DSE for the following entry:

```
supportedFeatures: 1.3.6.1.4.1.30221.2.12.3
```

Indexing a changelog attribute

Steps

1. Use **`dsconfig`** to set attribute indexing on an attribute in the Changelog Backend.

   The following command enables the Changelog Backend and sets the backend to include all user attributes ("*") for ADD or MODIFY operations using the `changelog-include-attribute` property. The `changelog-deleted-entry-include-attribute` property is set to all attributes ("*") to specify a set of attribute types that should be included in a changelog entry for DELETE operations. Attributes specified in this list will be recorded in the `deletedEntryAttrs` attribute on the changelog entry when an entry is deleted. The attributes `displayName` and `employeeNumber` are indexed using the `index-include-attribute` property.

   ```
   $ bin/dsconfig set-backend-prop --backend-name changelog \
     --set "enabled:true" \
     --set "changelog-include-attribute:*" \
     --set "changelog-deleted-entry-include-attribute:*" \
     --set "index-include-attribute:displayName" \
     --set "index-include-attribute:employeeNumber"
   ```

2. Add another attribute to index using the **`dsconfig`** `--add` option, which adds the attribute to an existing configuration setting.

   ```
   $ bin/dsconfig set-backend-prop --backend-name changelog \
     --add "index-include-attribute:cn"
   ```

Excluding attributes from indexing

Steps

- Use **`dsconfig`** to set attribute indexing on all user attributes in the Changelog Backend. The following command includes all user attributes except the description and location attributes.

  ```
  $ bin/dsconfig set-backend-prop --backend-name changelog \
    --set "index-include-attribute:*" \
    --set "index-exclude-attribute:description \
    --set "index-exclude-attribute:location
  ```

Tracking virtual attribute changes in the LDAP changelog

About this task

While the LDAP Changelog primarily tracks changes to real attributes, it can also provide information about virtual attributes included in created, updated, or deleted entries. The `include-virtual-attributes` property controls the virtual attribute information to include in changelog entries, and this property may include any or all of the following values:

add-attributes

Indicates that changelog entries for add operations should include a `ds-changelog-virtual-attributes` attribute that lists the virtual attribute values generated for the entry at the time it was created.

deleted-entry-attributes

Indicates that changelog entries for delete operations should include a `ds-changelog-virtual-attributes` attribute that lists the virtual attribute values generated for the entry at the time it was deleted.

before-and-after-values

Indicates that changelog entries for modify and modify DN operations should include `ds-changelog-before-virtual-values` and `ds-changelog-after-virtual-values` attributes that contain the values of virtual attributes that have been updated.

key-attribute-values

Indicates that changelog entries should include a `ds-changelog-entry-key-virtual-values` attribute that holds the values for any virtual attributes included in the set of key attributes, as defined by the `changelog-include-key-attribute` property.

The add-attributes and before-and-after-values options are selected by default.

Steps

- Use **`dsconfig`** to enable virtual attribute change tracking in the LDAP Changelog.

  The following command enables the LDAP changelog and sets `include-virtual-attributes` to `add-attributes`, which indicates that virtual attribute be included in the set of attributes listed for an add operation. The `delete-entry-attributes` option indicates that virtual attributes should be included in the set of deleted entry attributes listed for a delete operation. The `before-and-after-values` option indicates that virtual attributes should be included in the set of before and after values for attributes targeted by the changes. The `key-attribute-values` option indicates that virtual attributes should be included in the set of entry key attribute values.

```
$ bin/dsconfig set-backend-prop --backend-name "changelog" \
  --set "enabled:true" \
  --set "include-virtual-attributes:add-attributes" \
  --set "include-virtual-attributes:deleted-entry-attributes" \
  --set "include-virtual-attributes: before-and-after-values" \
  --set "include-virtual-attributes: key-attribute-values"
```

Managing Access Control

The PingDirectory Server provides a fine-grained access control model to ensure that users are able to access the information they need, but are prevented from accessing information that they should not be allowed to see. It also includes a privilege subsystem that provides even greater flexibility and protection in many key areas.

This chapter presents the access control model and provides examples that illustrate the use of key access control functionality.

Overview of access control

The access control model uses access control instructions (ACIs), which are stored in the `aci` operational attribute, to determine what a user or a group of users can do with a set of entries, down to the attribute level. The operational attribute can appear on any entry and affects the entry or any subentries within that branch of the directory information tree (DIT).

Access control instructions specifies four items:

- **Resources**. *Resources* are the targeted items or objects that specifies the set of entries and/or operations to which the access control instruction applies. For example, you can specify access to certain attributes, such as the `cn` or `userPassword` password.
- **Name**. *Name* is the descriptive label for each access control instruction. Typically, you will have multiple access control instructions for a given branch of your DIT. The access control name helps describe its purpose. For example, you can configure an access control instructions labelled "ACI to grant full access to administrators."
- **Clients**. *Clients* are the users or entities to which you grant or deny access. You can specify individual users or groups of users using an LDAP URL. For example, you can specify a group of administrators using the LDAP URL: `groupdn="ldap:///cn=admins,ou=groups,dc=example,dc=com."`
- **Rights**. *Rights* are permissions granted to users or client applications. You can grant or deny access to certain branches or operations. For example, you can grant `read` or `write` permission to a `telephoneNumber` attribute.

Key access control features

The PingDirectory Server provides important access control features that provide added security for the Directory Server's entries.

Improved validation and security

The Directory Server provides an access control model with strong validation to help ensure that invalid ACIs are not allowed into the server. For example, the Directory Server ensures that all access control rules added over LDAP are valid and can be fully parsed. Any operation that attempts to store one or more invalid ACIs are rejected. The same validation is applied to ACIs contained in data imported from an LDIF file. Any entry containing a malformed `aci` value will be rejected.

As an additional level of security, the Directory Server examines and validates all ACIs stored in the data whenever a backend is brought online. If any malformed ACIs are found in the backend, then the server generates an administrative alert to notify administrators of the problem and places itself in lockdown mode. While in lockdown mode, the server only allows requests from users who have the `lockdown-mode` privilege. This action allows administrators to correct the malformed ACI while ensuring that no sensitive data is inadvertently exposed due to an access control instruction not being enforced. When the problem has been corrected, the administrator can use the **`leave-lockdown-mode`** tool or restart the server to allow it to resume normal operation.

Global ACIs

Global ACIs are a set of ACIs that can apply to entries anywhere in the server (although they can also be scoped so that they only apply to a specific set of entries). They work in conjunction with access control rules stored in user data and provide a convenient way to define ACIs that span disparate portions of the DIT.

In the PingDirectory Server, global ACIs are defined within the server configuration, in the `global-aci` property of configuration object for the access control handler. They can be viewed and managed using configuration tools like **`dsconfig`** and the Administrative Console.

The global ACIs available by default in the PingDirectory Server include:

- Allow anyone (including unauthenticated users) to access key attributes of the root DSE, including: `namingContexts`, `subschemaSubentry`, `supportedAuthPasswordSchemes`, `supportedControl`, `supportedExtension`, `supportedFeatures`, `supportedLDAPVersion`, `supportedSASLMechanisms`, `vendorName`, and `vendorVersion`.
- Allow anyone (including unauthenticated users) to access key attributes of the subschema subentry, including: `attributeTypes`, `dITContentRules`, `dITStructureRules`, `ldapSyntaxes`, `matchingRules`, `matchingRuleUse`, `nameForms`, and `objectClasses`.
- Allow anyone (including unauthenticated users) to include the following controls in requests made to the server: authorization identity request, manage DSA IT, password policy, real attributes only, and virtual attributes only.
- Allow anyone (including unauthenticated users) to request the following extended operations: get symmetric key, password modify request, password policy state, StartTLS, and Who Am I?

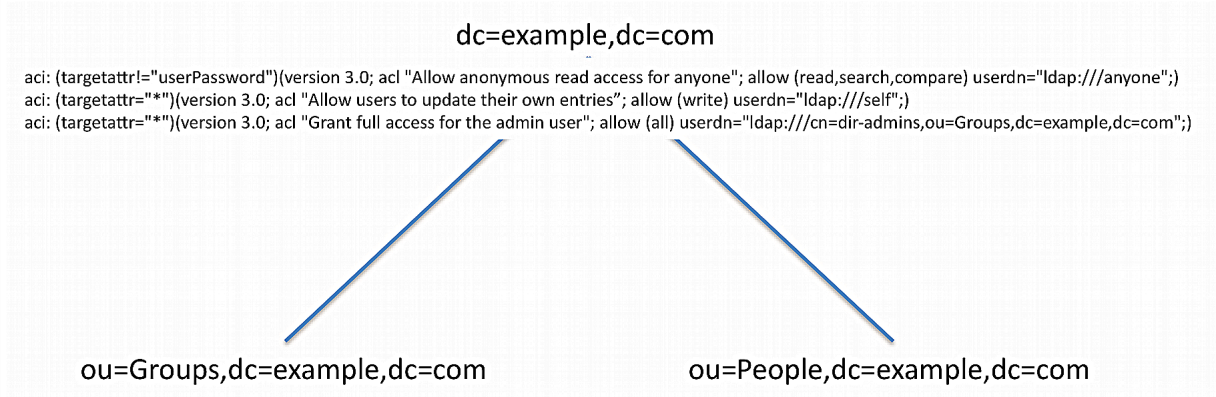
Access controls for public or private backends

The PingDirectory Server classifies backends as either public or private, depending on their intended purpose. A private backend is one whose content is generated by the Directory Server itself (for example, the root DSE, monitor, and backup backends), is used in the operation of the server (for example, the configuration, schema, task, and trust store backends), or whose content is maintained by the server (for example, the LDAP changelog backend). A public backend is intended to hold user-defined content, such as user accounts, groups, application data, and device data.

The PingDirectory Server access control model also supports the distinction between public backends and private backends. Many private backends do not allow writes of any kind from clients, and some of the private backends that do allow writes only allow changes to a specific set of attributes. As a result, any access control instruction intended to permit or restrict access to information in private backends should be defined as global ACIs, rather than attempting to add those instructions to the data for that private backend.

General format of the access control rules

Access control instructions (ACIs) are represented as strings that are applied to one or more entries within the Directory Information Tree (DIT). Typically, an ACI is placed on a subtree, such as `dc=example,dc=com`, and applies to that base entry and all entries below it in the tree. The Directory Server iterates through the DIT to compile the access control rules into an internally-used list of denied and allowed targets and their permissable operations. When a client application, such as **`ldapsearch`**, enters a request, the Directory Server checks that the user who binds with the server has the necessary access rights to the requested search targets. ACIs are cumulatively applied, so that a user who may have an ACI at an entry, may also have other access rights available if ACIs are defined higher in the DIT and are applicable to the user. In most environments, ACIs are defined at the root of a main branch or a subtree, and not on individual entries unless absolutely required.



MY TITLE ACI

An access control rule has a basic syntax as follows:

```
aci : (targets) (version 3.0; acl "name";
  permissions
  bind rules
  ;)
```

Access Control Components

Access Control Component	Description
targets	Specifies the set of entries and/or attributes to which an access control rule applies. Syntax: *(target keyword* = || *!= expression)*
name	Specifies the name of the ACI.
permissions	Specifies the type of operations to which an access control rule might apply. Syntax: *allow||deny (permission)*
bind rules	Specifies the criteria that indicate whether an access control rule should apply to a given requestor. Syntax: *bind rule keyword* = ||*!= expression;*. The bind rule syntax requires that it be terminated with a ";".

Summary of access control keywords

This section provides an overview of the keywords supported for use in the PingDirectory Server access control implementation.

Targets

A target expression specifies the set of entries and/or attributes to which an access control rule applies. The *keyword* specifies the type of target element. The *expression* specifies the items that is targeted by the access control rule. The operator is either the equal ("=") or not-equal ("!="). Note that the "!=" operator cannot be used with `targattrfilters` and `targetscope` keywords. For specific examples on each target keyword, see the section *Working with Targets*.

```
(keyword [=||!=]expression)
```

The following keywords are supported for use in the target portion of ACIs:

Summary of Access Control Target Keywords

Target Keyword	Description	Wildcards
extop	Specifies the OIDs for any extended operations to which the access control rule should apply.	No
target	Specifies the set of entries, identified using LDAP URLs, to which the access control rule applies.	**Yes**
targattrfilters	Identifies specific attribute values based on filters that may be added to or removed from entries to which the access control rule applies.	**Yes**
targetattr	Specifies the set of attributes to which the access control rule should apply.	**Yes**
targetcontrol	Specifies the OIDs for any request controls to which the access control rule should apply.	No

Target Keyword	Description	Wildcards
targetfilter	Specifies one or more search filters that may be used to indicate the set of entries to which the access control should apply.	**Yes**
targetscope	Specifies the scope of entries, relative to the defined target entries or the entry containing the ACI fi there is no target, to which the access control rule should apply.	No

Permissions

Permissions indicate the types of operations to which an access control rule might apply. You can specify if the user or group of users are allowed or not allowed to carry out a specific operation. For example, you would grant read access to a targeted entry or entries using "allow (read)" permission. Or you can specifically deny access to the target entries and/or attributes using the "deny (read)" permission. You can list out multiple permissions as required in the ACI.

```
allow (permission1 ...,
permission2
,...permissionN)
```

```
deny (permission1 ...,
permission2
,...permissionN)
```

The following keywords are supported for use in the permissions portion of ACIs:

Summary of Access Control Permissions

Permission	Description
add	Indicates that the access control should apply to add operations.
compare	Indicates that the access control should apply to compare operations, as well as to search operations with a base-level scope that targets a single entry.
delete	Indicates that the access control should apply to delete operations.
export	Indicates that the access control should only apply to modify DN operations in which an entry is moved below a different parent by specifying a new superior DN in the modify DN request. The requestor must have the `export` permission for operations against the entry's original DN. The requestor must have the `import` permission for operations against the entry's new superior DN. For modify DN operations that merely alter the RDN of an entry but keeps it below the same parent (i.e., renames the entry), only the `write` permission is required. This is true regardless of whether the entry being renamed is a leaf entry or has subordinate entries.
import	See the description for the `export` permission.
proxy	Indicates that the access control rule should apply to operations that attempt to use an alternate authorization identity (for example, operations that include a proxied authorization request control, an intermediate client request control with an alternate authorization identity, or a client that has authenticated with a SASL mechanism that allows an alternate authorization identify to be specified).

Permission	Description
read	Indicates that the access control rule should apply to search result entries returned by the server.
search	Indicates that the access control rule should apply to search operations with a non-base scope.
selfwrite	Indicates that the access control rule should apply to operations in which a user attempts to add or remove his or her own DN to the values for an attribute (for example, whether users may add or remove themselves from groups).
write	Indicates that the access control rule should apply to modify and modify DN operations.
all	An aggregate permission that includes all other permissions except “proxy.” This is equivalent to providing a permission of “add, compare, delete, read, search, selfwrite, write, export, and import.”

Bind rules

The Bind Rules indicate whether an access control rule should apply to a given requester. The syntax for the target keyword is shown below. The *keyword* specifies the type of target element. The expression specifies the items that is targeted by the access control rule. The operator is either equals ("=") or not-equals ("!="). The semi-colon delimiter symbol (";") is required after the end of the final bind rule.

```
keyword [=||!= ] expression;
```

Multiple bind rules can be combined using Boolean operations (AND, OR, NOT) for more access control precision. The standard Boolean rules for evaluation apply: innermost to outer parentheses first, left to right expressions, NOT before AND or OR. For example, an ACI that includes the following bind rule targets all users who are not `uid=admin,dc=example,dc=com` and use simple authentication.

```
(userdn!="ldap:///uid=admin,dc=example,dc=com" and authmethod="simple");
```

The following bind rule targets the `uid=admin,dc=example,dc=com` and authenticates using SASL EXTERNAL or accesses the server from a loopback interface.

```
(userdn="ldap:///uid=admin,dc=example,dc=com and (authmethod="SSL" or
 ip="127.0.0.1"));
```

The following keywords are supported for use in the bind rule portion of ACIs:

Summary of Bind Rule Keywords

Bind Rule Keyword	Description

authmethod	Indicates that the requester's authentication method should be taken into account when determining whether the access control rule should apply to an operation. Wildcards are not allowed in this expression. The keyword's syntax is as follows:
	
	authmethod = *method*
	
	where *method* is one of the following representations:
	▪ none
	▪ simple. Indicates that the client is authenticated to the server using a bind DN and password.
	▪ ssl. Indicates that the client is authenticated with an SSL/TLS certificate (for example, via SASL EXTERNAL), and not just over a secure connection to the server.
	▪ sasl {sasl_mechanism}. Indicates that the client is authenticated to the server using a specified SASL Mechanism.
	
	The following example allows users who authenticate with an SSL/TLS certificate (for example., via SASL EXTERNAL) to update their own entries:
	
	`aci: (targetattr="*")`
	`  (version 3.0; acl "Allow users to update their own entries";`
	`   allow (write) (userdn="ldap:///self" and authmethod="ssl");)`

dayofweek	Indicates that the day of the week should be taken into account when determining whether the access control rule should apply to an operation. Wildcards are not allowed in this expression. Multiple day of week values may be separated by commas. The keyword's syntax is as follows:
	
	dayofweek = *day1, day2, ...*
	
	where *day* is one of the following representations:
	▪ sun
	▪ mon
	▪ tues
	▪ wed
	▪ thu
	▪ fri
	▪ sat
	
	The following example allows users who authenticate with an SSL/TLS certificate (for example., via SASL EXTERNAL) on weekdays to update their own entries:
	
	`aci: (targetattr="*")`
	`  (version 3.0; acl "Allow users to update their own entries";`
	`   allow (write) (dayofweek!="sun,sat" and userdn="ldap:///`
	`self"`
	`   and authmethod="ssl");)`

Bind Rule Keyword	Description

dns	Indicates that the requester's DNS-resolvable host name should be taken into account when determining whether the access control rule should apply to an operation. Wildcards are allowed in this expression. Multiple DNS patterns may be separated by commas. The keyword's syntax is as follows:
	
	dns = *dns-host-name*
	
	The following example allows users on host name `server.example.com` to update their own entries:
	
	`aci: (targetattr="*")`
	`  (version 3.0; acl "Allow users to update their own entries";`
	`  allow (write) (dns="server.example.com" and userdn="ldap:///`
	`self");)`

groupdn	Indicates that the requester's group membership should be taken into account when determining whether the access control rule should apply to any operation. Wildcards are not allowed in this expression.
	
	groupdn [= || !=] "ldap:///groupdn [|| ldap:///groupdn] ..."
	
	The following example allows users in the managers group to update their own entries:
	
	`aci: (targetattr="*")`
	`  (version 3.0; acl "Allow users to update their own entries";`
	`  allow (write)`
	`  (groupdn="ldap:///`
	`cn=managers,ou=groups,dc=example,dc=com");)`

Bind Rule Keyword	Description

ip	Indicates that the requester's IP address should be taken into account when determining whether the access control rule should apply to an operation. Wildcards are allowed in this expression. Multiple IP address patterns may be separated by commas. The keyword's syntax is as follows:
	
	ip `[ = || != ]` *ipAddressList*
	
	where *ipAddressList* is one of the following representations:
	▪ A specific IPv4 address: 127.0.0.1
	▪ An IPv4 address with wildcards to specify a subnetwork: 127.0.0.*
	▪ An IPv4 address or subnetwork with subnetwork mask: 123.4.5.0+255.255.255.0
	▪ An IPv4 address range using CIDR notation: 123.4.5.0/24
	▪ An IPv6 address as defined by RFC 2373.
	
	The following example allows users on 10.130.10.2 and localhost to update their own entries:
	
	`aci: (targetattr="*")`
	`  (version 3.0; acl "Allow users to update their own entries";`
	`   allow (write) (ip="10.130.10.2,127.0.0.1" and`
	` userdn="ldap:///self");)`

oauthscope	Indicates that the scopes associated with any OAuth 2.0 access token presented by a SCIMv2 client should be taken into account when determining whether the access control rule should apply to an operation. The keyword's syntax is as follows:
	
	`oauthscope [ = || != ] "scopeIdentifier"`
	
	where *scopeIdentifier* is one of the following:
	▪ The name of a single scope to match. The name will be treated as case-sensitive.
	▪ A substring assertion that contains one or more asterisks as wildcards.
	▪ A single asterisk by itself, which will match any scope.
	
	The following example will grant all rights to any client that presented an OAuth 2.0 token that is associated with the "scim_admin" scope:
	
	`aci: (targetattr="*")`
	`  (version 3.0; acl "Full rights for users with the scim_admin`
	` OAuth 2.0 scope";`
	`    allow (all) oauthscope="scim_admin";)`

Bind Rule Keyword	Description

timeofday	Indicates that the time of day should be taken into account when determining whether the access control rule should apply to an operation. Wildcards are not allowed in this expression. The keyword's syntax is as follows:
	
	`timeofday [ = || != || >= || > || <= || < ] time`
	
	where *time* is one of the following representations:
	
	▪ 4-digit 24-hour time format (0000 to 2359, where the first two digits represent the hour of the day and the last two represent the minute of the hour)
	▪ Wildcards are not allowed in this expression
	
	The following example allows users to update their own entries if the request is received before 12 noon.
	
	`aci: (targetattr="*")`
	`  (version 3.0; acl "Allow users who authenticate before noon`
	`    to update their own entries";`
	`    allow (write) (timeofday<1200 and userdn="ldap:///self"`
	`    and authmethod="simple");)`

oauthscope	Indicates that the scopes associated with any OAuth 2.0 access token presented by a SCIMv2 client should be taken into account when determining whether the access control rule should apply to an operation. The keyword's syntax is as follows:
	
	`oauthscope [ = || != ] "scopeIdentifier"`
	
	where *scopeIdentifier* is one of the following:
	
	▪ The name of a single scope to match. The name will be treated as case-sensitive.
	▪ A substring assertion that contains one or more asterisks as wildcards.
	▪ A single asterisk by itself, which will match any scope.
	
	The following example will grant all rights to any client that presented an OAuth 2.0 token that is associated with the "scim_admin" scope:
	
	`aci: (targetattr="*")`
	`  (version 3.0; acl "Full rights for users with the scim_admin`
	`OAuth 2.0 scope";`
	`    allow (all) oauthscope="scim_admin";)`

Bind Rule Keyword	Description

userattr	Indicates that the requester's relation to the value of the specified attribute should be taken into account when determining whether the access control rule should apply to an operation. A bindType value of `USERDN` indicates that the target attribute should have a value which matches the DN of the authenticated user. A bindType value of `GROUPDN` indicates that the target attribute should have a value which matches the DN of a group in which the authenticated user is a member. A bindType value of `LDAPURL` indicates that the target attribute should have a value that is an LDAP URL whose criteria matches the entry for the authenticated user. Any value other than `USERDN`, `GROUPDN`, or `LDAPURL` is expected to be present in the target attribute of the authenticated user's entry. The keyword's syntax is as follows:
	
	`userattr = attrName# [ bindType || attrValue ]`
	
	where:
	▪ *attrName* = name of the attribute for matching
	▪ *bindType* = USERDN, GROUPDN, LDAPURL
	▪ *attrValue* = an attribute value. Note that the *attrVALUE* of the attribute must match on both the bind entry and the target of the ACI.
	
	The following example allows a manager to change employee's entries. If the bind DN is specified in the *manager* attribute of the targeted entry, the bind rule is evaluated to TRUE.
	
	`aci: (targetattr="*")`
	`  (version 3.0; acl "Allow a manager to change employee entries";`
	`   allow (write) userattr="manager#USERDN";)`
	
	The following example allows any member of a group to change employee's entries. If the bind DN is a member of the group specified in the *allowEditors* attribute of the targeted entry, the bind rule is evaluated to TRUE.
	
	`aci: (targetattr="*")`
	`  (version 3.0; acl "Allow allowEditors to change employee entries";`
	`   allow (write) userattr="allowEditors#GROUPDN";)`
	
	The following example allows a user's manager to edit that user's entry and any entries below the user's entry up to two levels deep. You can specify up to five levels (0, 1, 2, 3, 4) below the targeted entry, with zero (0) indicating the targeted entry.
	
	`aci: (targetattr="*")`
	`  (version 3.0; acl "Allow managers to change employees entries two levels below";`
	`   allow (write) userattr="parent[0,1,2].manager#USERDN";)`
	
	The following example allows any member of the engineering department to update any other member of the engineering department at or below the specified ACI.
	
	`aci: (targetattr="*")`
	`  (version 3.0; acl "Allow any member of Eng Dept to update any other member of the`
	`   enginering department at or below the ACI";`
	`   allow (write) userattr="department#ENGINEERING";)`
	
	The following example allows an entry to be updated by any user whose entry matches the criteria defined in the LDAP URL contained in the `allowedEditorCriteria` attribute of the target entry.
	
	`aci: (targetattr="*")`
	`  (version 3.0; acl "Allow a user that matches the filter to change entries";`

Bind Rule Keyword	Description

userdn	Indicates that the user's DN should be taken into account when determining whether the access control rule should apply to an operation. The keyword's syntax is as follows:
	
	`userdn [ = || != ] "ldap:///value [ || "ldap:///value ..."]`
	
	where *value* is one of the following representations:
	
	▪ The DN of the target user
	▪ A value of `anyone` to match any client, including unauthenticated clients.
	▪ A value of `all` to match any authenticated client.
	▪ A value of `parent` to match the client authenticated as the user defined in the immediate parent of the target entry.
	▪ A value of `self` to match the client authenticated as the user defined in the target entry.
	
	If the value provided is a DN, then that DN may include wildcard characters to define patterns. A single asterisk will match any content within the associated DN component, and two consecutive asterisks may be used to match zero or more DN components.
	
	The following example allows users to update their own entries:
	
	`aci: (targetattr="*")`
	`  (version 3.0; acl "Allow users to update their own entries";`
	`  allow (write) userdn="ldap:///self";)`

Access token validators

Access token validators verify the tokens that HTTP client applications submit when they request access to protected resources and associate each token with an identity stored in the directory server.

To authenticate to PingDirectory Server's HTTP services, clients use OAuth 2 bearer token authentication to present an access token in the HTTP Authorization request header. To process the incoming access tokens, PingDirectory Server uses access token validators, which determine whether to accept an access token and translate it into a set of properties, called *claims*, which PingDirectory Server's HTTP services use to make access control decisions.

Most access tokens identify a user as its subject using the sub claim. Access token validators can retrieve the token subject's attributes from the directory using an identity mapper, which correlates the access token subject to an LDAP entry.

Access token validators are used by the following services:

- Directory REST API
- SCIM 2
- Delegated Admin
- Consent API

In addition, PingDirectory Server can be configured to accept access tokens provided by LDAP clients using the OAUTHBEARER SASL authentication method.

About access token validator processing

You can configure any number of access token validators for PingDirectory Server.

Each access token validator possesses an evaluation order index, an integer that determines its processing priority when multiple access token validators are configured. Lower values are processed before higher values.

1. If an incoming HTTP request contains an access token, the token is sent to the access token validator with the lowest evaluation order index.
2. The access token validator validates the access token. Validation logic varies by access token validator type, but the validator generally verifies the following information:
   - A trusted source issued the token.
   - The token is not expired.
3. If the access token contains a subject, the access token validator uses its identity mapper to find a matching LDAP entry.
4. If the access token validator is unable to validate the access token, it passes the token to the access token validator with the next lowest evaluation order index, and the previous two steps are repeated.
5. HTTP request processing continues, and the policy request is sent to the HTTP service, such as the Directory REST API, for further evaluation.
6. Using either the access token claims parsed by the access token validator or the LDAP entry found by the identity mapper, the HTTP service determines whether the request should be accepted and which access control rules should be applied. This access control behavior varies by each HTTP service.

> ⓘ **Note:**
>
> Access tokens issued using the OAuth 2 client credentials grant type are issued directly to a client and do not contain a subject. Such tokens cannot be accepted by PingDirectory Server.

Access token validator types

PingDirectory Server works with a variety of access token validators.

PingFederate access token validator

To verify the access tokens that a PingFederate authorization server issues, the PingFederate access token validator uses HTTP to submit the tokens to PingFederate Server's token introspection endpoint.

This step allows the authorization server to determine whether a token is valid.

> ⓘ **Note:**
>
> Access tokens issued using the OAuth 2 client credentials grant type are issued directly to a client and do not contain a subject. Such tokens cannot be accepted by PingDirectory Server.

Because this step requires an outgoing HTTP request to the authorization server, the PingFederate access token validator might perform slower than other access token validator types. The validation result is guaranteed to be current, which is an important consideration if the authorization server permits the revocation of access tokens.

Before attempting to use a PingFederate access token validator, create a client that represents the access token validator in the PingFederate configuration. This client must use the Access Token Validation grant type.

Example configuration

In PingFederate, create a client with the following properties:

- Client ID: Ping Identity
- Client authentication: Client Secret
- Allowed grant types: Access Token Validation

Take note of the client secret that is generated for the client, and use the PingDirectory Server's **`dsconfig`** command to create an access token validator, as shown.

```
# Create an identity mapper that expects the token subject to be a uid
dsconfig create-identity-mapper \
 --mapper-name "User ID Identity Mapper" \
 --type exact-match \
 --set enabled:true \
 --set match-attribute:uid \
 --set match-base-dn:ou=people,dc=example,dc=com
# Change the host name and port below, as needed
dsconfig create-external-server \
  --server-name "PingFederate External Server" \
  --type http \
  --set base-url:https://example.com:9031
# Create the Access Token Validator
dsconfig create-access-token-validator \
  --validator-name "PingFederate Access Token Validator" \
  --type ping-federate \
  --set enabled:true \
  --set "authorization-server:PingFederate External Server" \
  --set client-id:PingDataGovernance \
  --set "client-secret:<client secret>"
  --set evaluation-order-index:2000
  --set "identity-mapper:User ID Identity Mapper"
```

Replace `<client secret>` with the client secret value generated by the PingFederate client.

JWT access token validator

The JWT access token validator verifies access tokens that are encoded in JSON Web Token (JWT) format, which can be signed in JSON web signature (JWS) format or signed and encrypted in JSON web encryption (JWE) format.

The JWT access token validator inspects the JWT token without presenting it to an authorization server for validation. Because the JWT access token validator does not make a token introspection request for every access token that it processes, it performs faster than the PingFederate access token validator. The access token is self-validated however, so the JWT access token validator cannot determine whether the token has been revoked.

Supported JWS/JWE features

For signed tokens, the JWT access token validator supports the following JWT web algorithm (JWA) types:

- RS256
- RS384
- RS512
- ES256
- ES384
- ES512

For encrypted tokens, the JWT access token validator supports the following key-encryption algorithms:

- RSA-OAEP
- ECDH-ES
- ECDH-ES+A128KW
- ECDH-ES+A192KW
- ECDH-ES+A256KW

For encrypted tokens, the JWT access token validator supports the following content-encryption algorithms:

- A128CBC-HS256
- A192CBC-HS384
- A256CBC-HS512

The JWT access token validator configuration defines three *allow lists* for the JWS/JWE signing and encryption algorithms that it will accept. You should customize these allow lists to reflect only the signing and encryption algorithms used by your access token issuer and no others. Doing so minimizes the access token validator's security threat surface.

Configure these allow lists using the following configuration properties:

- `allowed-signing-algorithm`

  Specifies the signing algorithms that the access token validator accepts.
- `allowed-key-encryption-algorithm`

  Specifies the key-encryption algorithms that the access token validator accepts.
- `allowed-content-encryption-algorithm`

  Specifies the content-encryption algorithms that the access token validator accepts.

Handling signed tokens
All access tokens the JWT access token validator handles must be cryptographically signed by the token issuer. The JWT access token validator validates a token's signature using a public signing key provided by the issuer.

Steps

- Configure the JWT access token validator with the issuer's public signing key in one of two ways:
  - Store the public key as a trusted certificate in PingDirectory Server's local configuration using the `trusted-certificate` property.
  - Provide the issuer's JWKS (JSON Web Key Set) endpoint using the `jwks-endpoint-path` property. The JWT access token validator then retrieves the issuer's public keys when it initializes. This method ensures that the JWT access token validator uses updated copies of the issuer's public keys.

Example: Use a locally configured trusted certificate
The following example configures a JWT access token validator to use a locally stored public signing certificate to validate access token signatures. The signing certificate is assumed to have been obtained out of band and must be a PEM-encoded X.509v3 certificate.

```
# Create an identity mapper that expects the token subject to be a uid
dsconfig create-identity-mapper \
 --mapper-name "User ID Identity Mapper" \
 --type exact-match \
 --set enabled:true \
 --set match-attribute:uid \
 --set match-base-dn:ou=people,dc=example,dc=com

# Add the public signing certificate to the server configuration
dsconfig create-trusted-certificate \
 --certificate-name "JWT Signing Certificate" \
 --set "certificate</path/to/signing-certificate.pem"

# Create the Access Token Validator
dsconfig create-access-token-validator \
 --validator-name "JWT Access Token Validator" \
 --type jwt \
 --set enabled:true \
 --set evaluation-order-index:1000 \
 --set allowed-signing-algorithm:RS256 \
 --set "trusted-certificate:JWT Signing Certificate"
 --set "identity-mapper:User ID Identity Mapper"
```

Example: Use the issuer's JWKS endpoint

The following example configures a JWT access token validator to retrieve public keys from a PingFederate authorization server's JWKS endpoint.

```
# Create an identity mapper that expects the token subject to be a uid
dsconfig create-identity-mapper \
 --mapper-name "User ID Identity Mapper" \
 --type exact-match \
 --set enabled:true \
 --set match-attribute:uid \
 --set match-base-dn:ou=people,dc=example,dc=com

# Change the host name and port below, as needed
dsconfig create-external-server \
 --server-name "PingFederate External Server" \
 --type http \
 --set base-url:https://example.com:9031

# Create the Access Token Validator
dsconfig create-access-token-validator \
 --validator-name "JWT Access Token Validator" \
 --type jwt \
 --set enabled:true \
 --set evaluation-order-index:1000 \
 --set allowed-signing-algorithm:RS256 \
 --set "authorization-server:PingFederate External Server" \
 --set jwks-endpoint-path:/ext/oauth/jwks
 --set "identity-mapper:User ID Identity Mapper"
```

Handling encrypted tokens

Optionally, you can configure the JWT access token validator to accept encrypted access tokens. To do this, you must configure the access token validator with a private/public key pair and provide the public key to the token issuer.

Steps

1. Create an encryption key pair.
2. Create the JWT access token validator.
3. Export the public encryption key from PingDataGovernance Server and provide it to your token issuer.

   Without this public encryption key, the issuer cannot encrypt tokens that can be decrypted by the JWT access token validator.

   You can run **`dsconfig`** to copy the public key to a file, or you can copy the value of the key pair's `certificate-chain` property in the Administrative Console.

Example

The following example configures a JWT access token validator to handle access tokens signed and encrypted using elliptic curve algorithms. For RSA signing and encryption algorithms, the configuration is very similar, but you would choose different values for the `allowed-signing-algorithm` and `allowed-encryption-algorithm` properties.

1. Create an encryption key pair.

```
# Create an encryption key pair
 dsconfig create-key-pair \
 --pair-name "JWT Elliptic Curve Encryption Key Pair" \
```

```
 --set key-algorithm:EC_256
```

2. Create the JWT access token validator.

```
# Create an identity mapper that expects the token subject to be a uid
dsconfig create-identity-mapper \
 --mapper-name "User ID Identity Mapper" \
 --type exact-match \
 --set enabled:true \
 --set match-attribute:uid \
 --set match-base-dn:ou=people,dc=example,dc=com

# Change the host name and port below, as needed
dsconfig create-external-server \
 --server-name "PingFederate External Server" \
 --type http \
 --set base-url:https://example.com:9031

# Create the Access Token Validator
dsconfig create-access-token-validator \
 --validator-name "JWT Access Token Validator" \
 --type jwt \
 --set enabled:true \
 --set evaluation-order-index:1000 \
 --set allowed-signing-algorithm:ES256 \
 --set "authorization-server:PingFederate External Server" \
 --set jwks-endpoint-path:/ext/oauth/jwks \
 --set "encryption-key-pair:JWT Elliptic Curve Encryption Key Pair" \
 --set allowed-key-encryption-algorithm:ECDH_ES
 --set "identity-mapper:User ID Identity Mapper"
```

3. Export the public encryption key from PingDataGovernance Server and provide it to your token issuer.

   The following command copies the key to a file.

```
 dsconfig get-key-pair-prop \
 --pair-name "JWT Elliptic Curve Encryption Key Pair" \
 --property certificate-chain \
 --no-prompt \
 --script-friendly > jwt-public-encryption-key.pem
```

Mock access token validator

A mock access token validator is a special access token validator type used for development or testing purposes.

A mock access token validator accepts arbitrary tokens without validating whether a trusted source issued them. This approach allows a developer or tester to make bearer token-authenticated requests without first setting up an authorization server.

Mock access tokens are formatted as plain-text JSON objects using standard JSON web token (JWT) claims.

Always provide the boolean `active` claim when creating a mock token. If this value is `true`, the token is accepted. If this value is `false`, the token is rejected.

If the `sub` claim is provided, a token owner lookup populates the `TokenOwner` policy request attribute, as with the other access token validator types.

The following example cURL command provides a mock access token in an HTTP request.

```
curl -k -X GET https://localhost:8443/scim/v2/Me -H 'Authorization:
 Bearer {"active": true, "sub":"user.3", "scope":"email profile",
 "client":"client1"}'
```

> ⓘ **Important:**
>
> Never use mock access token validators in a production environment because they do not verify whether a trusted source issued an access token.

Example configuration

The configuration for a mock access token validator resembles the configuration for a JWT access token validator. However, the JSON web signature (JWS) signatures require no configuration because mock tokens are not authenticated.

```
# Create an identity mapper that expects the token subject to be a uid
 dsconfig create-identity-mapper \
 --type exact-match \
 --set enabled:true \
 --set match-attribute:uid \
 --set match-base-dn:ou=people,dc=example,dc=com

# Create the Access Token Validator
dsconfig create-access-token-validator \
  --validator-name "Mock Access Token Validator" \
  --type mock --set enabled:true \
  --set evaluation-order-index:9999
  --set "identity-mapper:User ID Identity Mapper"
```

Third-party access token validator
Third-party access token validator

To create custom access token validators, use the Server SDK.

Working with targets

The following section presents a detailed look and examples of the target ACI keywords: `target`, `targetattr`, `targetfilter`, `targattrfilters`, `targetscope`, `targetcontrol`, and `extop`.

target

The `target` keyword indicates that the ACI should apply to one or more entries at or below the specified distinguished name (DN). The target DN must be equal or subordinate to the DN of the entry in which the ACI is placed. For example, if you place the ACI at the root of `ou=People,dc=example,dc=com`, you can target the DN, `uid=user.1,ou=People,dc=example,dc=com` within your ACI rule. The DN must meet the string representation specification of distinguished names, outlined in RFC 4514, and requires that special characters be properly escaped.

The `target` clause has the following format, where DN is the distinguished name of the entry or branch:

```
(target = ldap:///
                  DN
               )
```

For example, to target a specific entry, you would use a clause such as the following:

```
(target = ldap:///uid=john.doe,ou=People,dc=example,dc=com)
```

Note that, in general, specifying a target DN is not recommended. It is better to have the ACI defined in that entry and omit the `target` element altogether. For example, although you can have `(target="ldap:///uid=john.doe,ou=People,dc=example,dc=com)` in any of the `dc=example,dc=com` or `ou=People` entries, it is better for it to be defined in the `uid=john.doe` entry and not explicitly include the `target` element.

The expression allows for the "not equal" (!=) operator to indicate that all entries within the scope of the given branch that do NOT match the expression be targeted for the ACI. Thus, the following expression targets all entries within the subtree that do not match `uid=john.doe`.

```
(target != ldap:///uid=john.doe,ou=People,dc=example,dc=com)
```

The `target` keyword also supports the use of asterisk (*) characters as wildcards to match elements within the distinguished name. The following target expression matches all entries that contains and begins with "john.d, " so that entries like "`john.doe,ou=People,dc=example,dc=com`," and "`john.davies,ou=People,dc=example,dc=com`" would match.

```
(target = ldap:///uid=john.d*,ou=People,dc=example,dc=com)
```

The following target expression matches all entries whose DN begins with "john.d," and matches the `ou` attribute. Entries like "`john.doe,ou=People,dc=example,dc=com`," and "`john.davies,ou=asia-branch,dc=example,dc=com`" would match.

```
(target = ldap:///uid=john.d*,ou=*,dc=example,dc=com)
```

Another example of a complete ACI targets the entries in the `ou=People,dc=example,dc=com` branch and the entries below it, and grants the users the privilege to modify all of their user attributes within their own entries.

```
aci:(target="ldap:///ou=People,dc=example,dc=com")
  (targetattr="*")
  (version 3.0; acl "Allow all the ou=People branch to modify their own
 entries";
  allow (write) userdn="ldap:///self";)
```

targetattr

The `targetattr` keyword targets the attributes for which the access control instruction should apply. There are four general forms that it can take in the PingDirectory Server:

- **(targetattr="*")**. Indicates that the access control rule applies to all user attributes. Operational attributes will not automatically be included in this set.
- **(targetattr="+")**. Indicates that the access control rule applies to all operational attributes. User attributes will not automatically be included in this set.
- **(targetattr="attr1||attr2||attr3||...||attrN")**. Indicates that the access control rule applies only to the named set of attributes.
- **(targetattr!="attr1||attr2||attr3||...||attrN")**. Indicates that the access control rule applies to all user attributes except the named set of attributes. It will not apply to any operational attributes.

The targeted attributes can be classified as user attributes and operational attributes. User attributes define the actual data for that entry, while operational attributes provide additional metadata about the entry that can be used for informational purposes, such as when the entry was created, last modified and by whom. Metadata can also include attributes specifying which password policy applies to the user, or overrided default constraints like size limit, time limit, or look-through limit for that user.

The PingDirectory Server distinguishes between these two types of attributes in its access control implementation. The Directory Server does not automatically grant any access at all to operational attributes. For example, the following clause applies only to user attributes and not to operational attributes:

```
(targetattr="*")
```

You can also target multiple attributes in the entry. The following clause targets the common name (cn), surname (sn) and state (st) attribute:

```
(targetattr="cn||sn||st")
```

You can use the "+" symbol to indicate that the rule should apply to all operational attributes, as follows:

```
(targetattr="+")
```

To include all user and all operational attributes, you use both symbols, as follows:

```
(targetattr="*||+")
```

If there is a need to target a specific operational attribute rather than all operational attributes, then it can be specifically included in the values of the `targetattr` clause, as follows:

```
(targetattr="ds-rlim-size-limit")
```

Or if you want to target all user attributes and a specific operational attribute, then you can use them in the `targetattr` clause, as follows:

```
(targetattr="*||ds-rlim-size-limit")
```

The following ACIs are placed on the `dc=example,dc=com` tree and allows any user anonymous read access to all entries except the `userPassword` attribute. The second ACI allows users to update their own contact information. The third ACI allows the `uid=admin` user full access privileges to all user attributes in the `dc=example,dc=com` subtree.

```
aci: (targetattr!="userPassword")(version 3.0; acl "Allow anonymous
   read access for anyone"; allow (read,search,compare) userdn="ldap:///
anyone";)
aci: (targetattr="telephonenumber||street||homePhone||l||st")
  (version 3.0; acl "Allow users to update their own contact info";
   allow (write) userdn="ldap:///self";)
aci: (targetattr="*")(version 3.0; acl "Grant full access for the admin
 user";
  allow (all) userdn="ldap:///uid=admin,dc=example,dc=com";)
```

An important note must be made when assigning access to user and operational attributes, which can be outlined in an example to show the implications of the Directory Server not distinguishing between these attributes. It can be easy to inadvertently create an access control instruction that grants far more capabilities to a user than originally intended. Consider the following example:

```
aci: (targetattr!="uid||employeeNumber")
  (version 3.0; acl "Allow users to update their own entries";
    allow (write) userdn="ldap:///self";)
```

This instruction is intended to allow a user to update any attribute in his or her own entry with the exception of `uid` and `employeeNumber`. This ACI is a very common type of rule and seems relatively harmless on the surface, but it has very serious consequences for a Directory Server that does not distinguish between user attributes and operational attributes. It allows users to update operational attributes in their own entries, and could be used for a number of malicious purposes, including:

- A user could alter password policy state attributes to become exempt from password policy restrictions.
- A user could alter resource limit attributes and bypass size limit, time limit, and look-through-limit constraints.
- A user could add access control rules to his or her own entry, which could allow them to make their entry completely invisible to all other users including administrators granted full rights by access control rules, but excluding users with the `bypass-acl` privilege, allow them to edit any other attributes in their

own entry including those excluded by rules like `uid` and `employeeNumber` in the example above, or add, modify, or delete any entries below his or her own entry.

Because the PingDirectory Server does not automatically include operational attributes in the target attribute list, these kinds of ACIs do not present a security risk for it. Also note that users cannot add ACIs to any entries unless they have the `modify-acl` privilege.

Another danger in using the `(targetattr!="x")` pattern is that two ACIs within the same scope could have two different `targetattr` policies that cancel each other out. For example, if one ACI has `(targetattr!="cn||sn")` and a second ACI has `(targetattr!="userPassword")`, then the net effect is `(targetattr="*")`, because the first ACI inherently allows `userPassword`, and the second allows `cn` and `sn`.

targetfilter

The `targetfilter` keyword targets all attributes that match results returned from a filter. The `targetfilter` clause has the following syntax:

```
(targetfilter =
                    ldap_filter
                )
```

For example, the following clause targets all entries that contain "ou=engineering" attribue:

```
(targetfilter = "(ou=engineering)")
```

You can only specify a single filter, but that filter can contain multiple elements combined with the OR operator. The following clause targets all entries that contain "ou=engineering," "ou=accounting," and "ou=marketing."

```
(targetfilter = "(|(ou=engineering)(ou=accounting)(ou=marketing)")
```

The following example allows the user, `uid=eng-mgr`, to modify the `departmentNumber`, `cn`, and `sn` attributes for all entries that match the filter `ou=engineering`.

```
aci:(targetfilter="(ou=engineering)")
  (targetattr="departmentNumber||cn||sn")
  (version 3.0; acl "example"; allow (write)
   userdn="ldap:///uid=eng-mgr,dc=example,dc=com";)
```

targattrfilters

The `targattrfilters` keyword targets specific attribute *values* that match a filtered search criteria. This keyword allows you to set up an ACI that grants or denies permissions on an attribute value if that value meets the filter criteria. The `targattrfilters` keyword applies to individual values of an attribute, not to the whole attribute. The keyword also allows the use of wildcards in the filters.

The keyword clause has the following formats:

```
(target = "add=attr1:Filter1 && attr2:Filter2... && attrn:FilterN,
del=attr1:Filter1 && attr2:Filter2 ... && attrN:FilterN" )
```

where

- **add** represents the operation of adding an attribute value to the entry
- **del** represents the operation of removing an attribute value from the entry
- **attr1, attr2... attrN** represents the targeted attributes
- **filter1, filter2 ... filterN** represents filters that identify matching attribute values

The following conditions determine when the attribute must satisfy the filter:

- When adding or deleting an entry containing an attribute targeted a `targattrfilters` element, each value of that attribute must satisfy the corresponding filter.

- When modifying an entry, if the operation adds one or more values for an attribute targeted by a `targattrfilters` element, each value must satisfy the corresponding filter. If the operation deletes one or more values for a targeted attribute, each value must satisfy the corresponding filter.
- When replacing the set of values for an attribute targeted by a `targattrfilters` element, each value removed must satisfy the delete filters, and each value added must satisfy the add filters.

The following example allows any user who is part of the `cn=directory server admins` group to add the `soft-delete-read` privilege.

```
aci:(targattrfilter="add=ds-privilege-name:(ds-privilege-name=soft-delete-
read)")
  (version 3.0; acl "Allow members of the directory server admins group to
 grant the
   soft-delete-read privilege"; allow (write)
   groupdn="ldap:///cn=directory server admins,ou=group,dc=example,dc=com";)
```

targetscope

The `targetscope` keyword is used to restrict the scope of an access control rule. By default, ACIs use a subtree scope, which means that they are applied to the target entry (either as defined by the target clause of the ACI, or the entry in which the ACI is define if it does not include a target), and all entries below it. However, adding the `targetscope` element into an access control rule can restrict the set of entries to which it applies.

The following `targetscope` keyword values are allowed:

- **base**. Indicates that the access control rule should apply only to the target entry and not to any of its subordinates.
- **onelevel**. Indicates that the access control rule should apply only to entries that are the immediate children of the target entry and not to the target entry itself, nor to any subordinates of the immediate children of the target entry.
- **subtree**. Indicates that the access control rule should apply to the target entry and all of its subordinates. This is the default behavior if no `targetscope` is specified.
- **subordinate**. Indicates that the access control rule should apply to all entries below the target entry but not the target entry itself.

The following ACI targets all users to view the operational attributes (`supportedControl`, `supportedExtension`, `supportedFeatures`, `supportedSASLMechanisms`, `vendorName`, and `vendorVersion`) present in the root DSE entry. The `targetscope` is base to limit users to view only those attributes in the root DSE.

```
aci: (target="ldap:///")(targetscope="base")
    (targetattr="supportedControl||supportedExtension||
     supportedFeatures||supportedSASLMechanisms||vendorName||vendorVersion")
    (version 3.0; acl "Allow users to view Root DSE Operational Attributes";
     allow (read,search,compare) userdn="ldap:///anyone")
```

targetcontrol

The `targetcontrol` keyword is used to indicate whether a given request control can be used by those users targeted in the ACI. Multiple OIDs can be provided by separating them with the two pipe characters (optionally surrounded by spaces). Wildcards are not allowed when specifying control OIDs.

The following ACI example shows the controls required to allow an administrator to use and manage the Soft-Delete feature. The Soft Delete Request Control allows the user to soft-delete an entry, so that it could be undeleted at a later time. The Hard Delete Request Control allows the user to permanently remove an entry or soft-deleted entry. The Undelete Request Control allows the user to undelete a currently soft-deleted entry. The Soft-Deleted Entry Access Request Control allows the user to search for any soft-deleted entries in the server.

```
aci: (targetcontrol="1.3.6.1.4.1.30221.2.5.20||1.3.6.1.4.1.30221.2.5.22||
```

```
   1.3.6.1.4.1.30221.2.5.23||1.3.6.1.4.1.30221.2.5.24")
  (version 3.0; acl "Allow admins to use the Soft Delete Request Control,
   Hard Delete Request Control,Undelete Request Control, and
   Soft-deleted entry access request control";
   allow (read) userdn="ldap:///uid=admin,dc=example,dc=com";)
```

extOp

The `extop` keyword can be used to indicate whether a given extended request operation can be used. Multiple OIDs can be provided by separating them with the two pipe characters (optionally surrounded by spaces). Wildcards are not allowed when specifying extended request OIDs.

The following ACI allows the `uid=user-mgr` to use the Password Modify Request (i.e., OID=1.3.6.1.4.1.4203.1.11.1) and the StartTLS (i.e., OID=1.3.6.1.4.1.1466.20037) extended request OIDs.

```
aci:(extop="1.3.6.1.4.1.4203.1.11.1 || 1.3.6.1.4.1.1466.20037")
  (version 3.0; acl "Allows the mgr to use the Password Modify Request and
 StartTLS;
   allow(read) userdn="ldap:///uid=user-mgr,ou=people,dc=example,dc=com";)
```

Examples of common access control rules

This section provides a set of examples that demonstrate access controls that are commonly used in your environment. Note that to be able to alter access control definitions in the server, a user must have the `modify-acl` privilege as discussed later in this chapter.

Administrator access

The following ACI can be used to grant any member of the "`cn=admins,ou=groups,dc=example,dc=com`" group to add, modify and delete entries, reset passwords and read operational attributes such as `isMemberOf` and password policy state:

```
aci: (targetattr="+")(version 3.0; acl "Administrators can read, search or
 compare operational attributes";
allow (read,search,compare) groupdn="ldap:///
cn=admins,ou=groups,dc=example,dc=com";)
aci: (targetattr="*")(version 3.0; acl "Administrators can add, modify and
 delete entries";
allow (all) groupdn="ldap:///cn=admins,ou=groups,dc=example,dc=com";)
```

Anonymous and authenticated access

The following ACI allow anonymous read, search and compare on select attributes of `inetOrgPerson` entries while authenticated users can access several more. The authenticated user will inherit the privileges of the anonymous ACI. In addition, the authenticated user can change `userPassword`:

```
aci: (targetattr="objectclass || uid || cn || mail || sn || givenName")
(targetfilter="(objectClass=inetorgperson)")
(version 3.0; acl "Anyone can access names and email addresses of entries
 representing people";
allow (read,search,compare) userdn="ldap:///anyone";)
aci: (targetattr="departmentNumber || manager || isMemberOf")
(targetfilter="(objectClass=inetorgperson)")
(version 3.0; acl "Authenticated users can access these fields for entries
 representing people";
allow (read,search,compare) userdn="ldap:///all";)
aci: (targetattr="userPassword")(version 3.0; acl "Authenticated users can
 change password";
allow (write) userdn="ldap:///all";)
```

If no unauthenticated access should be allowed to the Directory Server, the preferred method for preventing unauthenticated, or anonymous access is to set the Global Configuration property `reject-unauthenticated-requests` to true.

Delegated access to a manager

The following ACI can be used to allow an employee's manager to edit the value of the employee's `telephoneNumber` attribute. This ACI uses the `userattr` keyword with a bind type of `USERDN`, which indicates that the target entry's manager attribute must have a value equal to the DN of the authenticated user:

```
aci: (targetattr="telephoneNumber")
(version 3.0; acl "A manager can update telephone numbers of her direct
 reports";
allow (read,search,compare,write) userattr="manager#USERDN";)
```

Proxy authorization

The following ACIs can be used to allow the application "`cn=OnBehalf,ou=applications,dc=example,dc=com`" to use the proxied authorization v2 control to request that operations be performed using an alternate authorization identity. The application user is also required to have the `proxied-auth` privilege as discussed later in this chapter:

```
aci: (version 3.0;acl "Application OnBehalf can proxy as another entry";
allow (proxy) userdn="ldap:///cn=OnBehalf,ou=applications,dc=example,dc=com";)
```

Validating ACIs before migrating data

Many directory servers allow for less restrictive application of their access control instructions, so that they accept invalid ACIs. For example, if Sun/Oracle encounters an access control rule that it cannot parse, then it will simply ignore it without any warning, and the server may not offer the intended access protection. Rather than unexpectedly exposing sensitive data, the PingDirectoryProxy Server rejects any ACIs that it cannot interpret, which ensures data access is properly limited as intended, but it can cause problems when migrating data with existing access control rules to a PingDirectoryProxy Server.

To validate an access control instruction, the PingDirectoryProxy Server provides a **`validate-acis`** tool in the `bin` directory (UNIX or Linux systems) or `bat` directory (Windows systems) that identifies any ACI syntax problems before migrating data. The tool can examine access control rules contained in either an LDIF file or an LDAP directory and write its result in LDIF with comments providing information about any problems that were identified. Each entry in the output will contain only a single ACI, so if an entry in the input contains multiple ACIs, then it may be present multiple times in the output, each time with a different ACI value. The entries contained in the output contains only ACI values, and all other attributes will be ignored.

Validating ACIs from a file

About this task

The **`validate-acis`** tool can process data contained in an LDIF file. It will ignore all attributes except `aci`, and will ignore all entries that do not contain the `aci` attribute, so any existing LDIF file that contains access control rules may be used.

Steps

1. Run the **`bin/validate-acis`** tool (UNIX or Linux systems) or **`bat\validate-acis`** (Win dows systems) by specifying the input file and output file. If the output file already exists, the existing contents will be re-written. If no output file is specified, then the results will be written to standard output.

```
$ bin/validate-acis --ldifFile test-acis.ldif --outputFile validated-
acis.ldif
```

```
# Processing complete # Total entries examined: 1
# Entries found with ACIs: 1
# Total ACI values found: 3
# Malformed ACI values found: 0
# Other processing errors encountered: 0
```

2. Review the results by opening the output file. For example, the `validated-acis.ldif` file that was generated in the previous step reads as follows:

```
# The following access control rule is valid
dn: dc=example,dc=com
aci: (targetattr!="userPassword")
  (version 3.0; acl "Allow anonymous read access for anyone";
    allow (read,search,compare) userdn="ldap:///anyone";)

# The following access control rule is valid
dn: dc=example,dc=com
aci: (targetattr="*")
  (version 3.0; acl "Allow users to update their own entries";
    allow (write) userdn="ldap:///self";)

# The following access control rule is valid
dn: dc=example,dc=com
aci: (targetattr="*")
  (version 3.0; acl "Grant full access for the admin user";
    allow (all) userdn="ldap:///uid=admin,dc=example,dc=com";)
```

3. If the input file has any malformed ACIs, then the generated output file will show what was incorrectly entered. For example, remove the quotation marks around `userPassword` in the original `test-acis.ldif` file, and re-run the command. The following command uses the `--onlyReportErrors` option to write any error messages to the output file only if a malformed ACI syntax is encountered.

```
$ bin/validate-acis --ldifFIle test-acis.ldif --outputFile validated-
acis.ldif \
  --onlyReportErrors
```

```
# Processing complete
# Total entries examined: 1
# Entries found with ACIs: 1
# Total ACI values found: 3
# Malformed ACI values found: 0
# Other processing errors encountered: 0
```

The output file shows the following message:

```
# The following access control rule is malformed or contains an unsupported
# syntax: The provided string '(targetattr!=userPassword)(version 3.0; acl
# "Allow anonymous read access for anyone"; allow (read,search,compare)
# userdn="ldap:///anyone";)' could not be parsed as a valid Access Control
# Instruction (ACI) because it failed general ACI syntax evaluation
dn: dc=example,dc=com
aci: (targetattr!=userPassword)
  (version 3.0; acl "Allow anonymous read access for anyone";
    allow (read,search,compare) userdn="ldap:///anyone";)
```

```
# The following access control rule is valid
dn: dc=example,dc=com
aci: (targetattr="*")
  (version 3.0; acl "Allow users to update their own entries";
    allow (write) userdn="ldap:///self";)

# The following access control rule is valid
dn: dc=example,dc=com
aci: (targetattr="*")
  (version 3.0; acl "Grant full access for the admin user";
    allow (all) userdn="ldap:///uid=admin,dc=example,dc=com";)
```

Validating ACIs in another Directory Server

About this task

The **validate-acis** tool also provides the ability to examine ACIs in data that exists in another Directory Server that you are planning to migrate to the PingDirectory Server. The tool helps to determine whether the Ping Identity Server accepts those ACIs.

Steps

- To use it in this manner, provide arguments that specify the address and port of the target Directory Server, credentials to use to bind, and the base DN of the subtree containing the ACIs to validate.

```
$ bin/validate-acis
```

```
# Processing complete # Total entries examined: 1
# Entries found with ACIs: 1
# Total ACI values found: 3
# Malformed ACI values found: 0
# Other processing errors encountered: 0
```

Migrating ACIs from Sun/Oracle to PingDirectory Server

This section describes the most important differences in access control evaluation between Sun/Oracle and the PingDirectory Server.

Support for macro ACIs

Sun/Oracle provides support for macros ACIs, making it possible to define a single ACI that can be used to apply the same access restrictions to multiple branches in the same basic structure. Macros ACIs are infrequently used and can cause severe performance degradation, so support for macros ACIs is not included in the PingDirectory Server. However, you can achieve the same result by simply creating the same ACIs in each branch.

Support for the roleDN bind rule

Sun/Oracle roles are a proprietary, non-standard grouping mechanism that provide little value over standard grouping mechanisms. The PingDirectory Server does not support DSEE roles and does not support the use of the `roleDN` ACI bind rule. However, the same behavior can be achieved by converting the DSEE roles to standard groups and using the `groupDN` ACI bind rule.

Targeting operational attributes

The Sun/Oracle access control model does not differentiate between user attributes and operational attributes. With Sun/Oracle, using `targetattr="*"` will automatically target both user and operational attributes. Using an exclusion list like `targetattr!="userPassword"` will automatically target all operational attributes in addition to all user attributes except `userPassword`. This behavior is responsible for several significant security holes in which users are unintentionally given access to operational

attributes. In some cases, it allows users to do things like exempt themselves from password policy restrictions.

In the PingDirectory Server, operational attributes are treated differently from user attributes and operational attributes are never automatically included. As such, `targetattr="*"` will target all user attributes but no operational attributes, and `targetattr!="userPassword"` will target all users attributes except `userPassword`, but no operational attributes. Specific operational attributes can be targeted by including the names in the list, like `targetattr="creatorsName||modifiersName"`. All operational attributes can be targeted using the `"+"` character. So, `targetattr="+"` targets all operational attributes but no user attributes and `targetattr="*||+"` targets all user and operational attributes.

Specification of global ACIs

Both DSEE and PingDirectory Server support global ACIs, which can be used to define ACIs that apply throughout the server. In servers with multiple naming contexts, this feature allows you to define a rule once as a global ACI, rather than needing to maintain an identical rule in each naming context.

In DSEE, global ACIs are created by modifying the root DSE entry to add values of the `aci` attribute. In the PingDirectory Server, global ACIs are managed with **`dsconfig`** referenced in the `global-aci` property of the Access Control Handler.

Defining ACIs for non-user content

In DSEE, you can write to the configuration, monitor, changelog, and tasks backends to define ACIs. In the PingDirectory Server, access control for private backends, like configuration, monitor, schema, changelog, tasks, encryption settings, backups, and alerts, should be defined as global ACIs.

Limiting access to controls and extended operations

DSEE offers limited support for restricting access to controls and extended operations. To the extent that it is possible to control such access with ACIs, DSEE defines entries with a DN such as "`oid={oid},cn=features,cn=config`" where `{oid}` is the OID of the associated control or extended operation. For example, the following DSEE entry defines ACIs for the persistent search control: "`oid=2.16.840.1.113730.3.4.3,cn=features,cn=config`".

In the PingDirectory Server, the "`targetcontrol`" keyword can be used to define ACIs that grant or deny access to controls. The "`extop`" keyword can be used to define ACIs that grant or deny access to extended operation requests.

Tolerance for malformed ACI values

In DSEE, if the server encounters a malformed access control rule, it simply ignores that rule without any warning. If this occurs, then the server will be running with less than the intended set of ACIs, which may prevent access to data that should have been allowed or, worse yet, may grant access to data that should have been restricted.

The PingDirectory Server is much more strict about the access control rules that it will accept. When performing an LDIF import, any entry containing a malformed or unsupported access control rule will be rejected. Similarly, any add or modify request that attempts to create an invalid ACI will be rejected. In the unlikely event that a malformed ACI does make it into the data, then the server immediately places itself in lockdown mode, in which the server terminates connections and rejects requests from users without the `lockdown-mode` privilege. Lockdown mode allows an administrator to correct the problem without risking exposure to user data.

ⓘ **Note:** Consider running the **`import-ldif`** tool with the `--rejectFile` option so that you can review any rejected ACIs.

About the privilege subsystem

In DSEE, only the root user is exempt from access control evaluation. While administrators can create ACIs that give "normal" users full access to any content, they can also create ACIs that would make some portion of the data inaccessible even to those users. In addition, some tasks can only be accomplished by the root user and you cannot restrict the capabilities assigned to that root user.

The PingDirectory Server offers a privilege subsystem that makes it possible to control the capabilities available to various users. Non-root users can be granted limited access to certain administrative capabilities, and restrictions can be enforced on root users. In addition, certain particularly risky actions (such as the ability to interact with the server configuration, change another user's password, impersonate another user, or shutdown and restart the server) require that the requester have certain privileges in addition to sufficient access control rights to process the operation.

Identifying unsupported ACIs

The PingDirectory Server provides a **`validate-acis`** tool that can be used to examine content in an LDIF file or data in another directory server (such as a DSEE instance) to determine whether the access control rules contained in that data are suitable for use in the PingDirectory Server instance. When migrating data from a DSEE deployment into a PingDirectory Server instance, the **`validate-acis`** tool should first be used to determine whether ACIs contained in the data are acceptable. If any problems are identified, then the data should be updated to correct or redefine the ACIs so that they are suitable for use in the PingDirectory Server.

For more information about using this tool, see *Validating ACIs Before Migrating Data*.

Working with privileges

In addition to the access control implementation, the PingDirectory Server includes a privilege subsystem that can also be used to control what users are allowed to do. The privilege subsystem works in conjunction with the access control subsystem so that privileged operations are only allowed if they are allowed by the access control configuration and the user has all of the necessary privileges.

Privileges can be used to grant normal users the ability to perform certain tasks that, in most other directories, would only be allowed for the root user. In fact, the capabilities extended to root users in the PingDirectory Server are all granted through privileges, so you can create a normal user account with the ability to perform some or all of the same actions as root users.

Administrators can also remove privileges from root users so that they are unable to perform certain types of operations. Multiple root users can be defined in the server with different sets of privileges so that the capabilities that they have are restricted to only the tasks that they need to be able to perform.

Available privileges

The following privileges are defined in the PingDirectory Server.

Summary of Privileges

Privilege	Description
audit-data-security	This privilege is required to initiate a data security audit on the server, which is invoked by the **`audit-data-security`** tool.
backend-backup	This privilege is required to initiate an online backup through the tasks interface. The server's access control configuration must also allow the user to add the corresponding entry in the tasks backend.
backend-restore	This privilege is required to initiate an online restore through the tasks interface. The server's access control configuration must also allow the user to add the corresponding entry in the tasks backend.

Privilege	Description
bypass-acl	This privilege allows a user to bypass access control evaluation. For a user with this privilege, any access control determination made by the server immediately returns that the operation is allowed. Note, however, that this does not bypass privilege evaluation, so the user must have the appropriate set of additional privileges to be able to perform any privileged operation (for example, a user with the `bypass-acl` privilege but without the `config-read` privilege is not allowed to access the server configuration).
bypass-pw-policy	This privilege allows a user entry to bypass password policy evaluation. This privilege is intended for cases where external synchronization might require passwords that violate the password validation rules. The privilege is also evaluated for bind operations, meaning password restrictions are also bypassed when binding as a user who has this privilege.
bypass-read-acl	This privilege allows the associated user to bypass access control checks performed by the server for bind, search, and compare operations. Access control evaluation may still be enforced for other types of operations.
config-read	This privilege is required for a user to access the server configuration. Access control evaluation is still performed and can be used to restrict the set of configuration objects that the user is allowed to see.
config-write	This privilege is required for a user to alter the server configuration. The user is also required to have the `config-read` privilege. Access control evaluation is still performed and can be used to restrict the set of configuration objects that the user is allowed to alter.
disconnect-client	This privilege is required for a user to request that an existing client connection be terminated. The connection is terminated through the disconnect client task. The server's access control configuration must also allow the user to add the corresponding entry to the tasks backend.
jmx-notify	This privilege is required for a user to subscribe to JMX notifications generated by the Directory Server. The user is also required to have the `jmx-read` privilege.
jmx-read	This privilege is required for a user to access any information provided by the Directory Server via the Java Management Extensions (JMX).
jmx-write	This privilege is required for a user to update any information exposed by the Directory Server via the Java Management Extensions (JMX). The user is also required to have the `jmx-read` privilege. Note that currently all of the information exposed by the server over JMX is read-only.
ldif-export	This privilege is required to initiate an online LDIF export through the tasks interface. The server's access control configuration must also allow the user to add the corresponding entry in the Tasks backend. To allow access to the Tasks backend, you can set up a global ACI that allows access to members of an Administrators group.
ldif-import	This privilege is required to initiate an online LDIF import through the tasks interface. The server's access control configuration must also allow the user to add the corresponding entry in the Tasks backend. To allow access to the Tasks backend, configure the global ACI as shown in the previous description of the **`ldif-export`** privilege.
lockdown-mode	This privilege allows the associated user to request that the server enter or leave lockdown mode, or to perform operations while the server is in lockdown mode.

Privilege	Description
modify-acl	This privilege is required for a user to add, modify, or remove access control rules defined in the server. The server's access control configuration must also allow the user to make the corresponding change to the `aci` operational attribute.
password-reset	This privilege is required for one user to be allowed to change another user's password. This privilege is not required for a user to be allowed to change his or her own password. The user must also have the access control instruction privilege to write the `userPassword` attribute to the target entry.
privilege-change	This privilege is required for a user to change the set of privileges assigned to a user, including the set of privileges, which are automatically granted to root users. The server's access control configuration must also allow the user to make the corresponding change to the `ds-privilege-name` operational attribute.
proxied-auth	This privilege is required for a user to request that an operation be performed with an alternate authorization identity. This privilege applies to operations that include the proxied authorization v1 or v2 control operations that include the intermediate client request control with a value set for the client identity field, or for SASL bind requests that can include an authorization identity different from the authentication identity.
server-restart	This privilege is required to initiate a server restart through the tasks interface. The server's access control configuration must also allow the user to add the corresponding entry in the tasks backend.
server-shutdown	This privilege is required to initiate a server shutdown through the tasks interface. The server's access control configuration must also allow the user to add the corresponding entry in the tasks backend.
soft-delete-read	This privilege is required for a user to access a soft-deleted-entry.
stream-values	This privilege is required for a user to perform a stream values extended operation, which obtains all entry DNs and/or all values for one or more attributes for a specified portion of the DIT.
unindexed-search	This privilege is required for a user to be able to perform a search operation in which a reasonable set of candidate entries cannot be determined using the defined index and instead, a significant portion of the database needs to be traversed to identify matching entries. The server's access control configuration must also allow the user to request the search.
update-schema	This privilege is required for a user to modify the server schema. The server's access control configuration must allow the user to update the operational attributes that contain the schema elements.

Privileges automatically granted to root users

The special abilities that root users have are granted through privileges. Privileges can be assigned to root users in two ways:

- By default, root users may be granted a specified set of privileges. Note that it is possible to create root users which are not automatically granted these privileges by including the `ds-cfg-inherit-default-root-privileges` attribute with a value of FALSE in the entries for those root users.
- Individual root users can have additional privileges granted to them, and/or some automatically-granted privileges may be removed from that user.

The set of privileges that are automatically granted to root users is controlled by the `default-root-privilege-name` property of the Root DN configuration object. By default, this set of privileges includes:

- audit-data-security
- backend-backup
- backend-restore
- bypass-acl
- config-read
- config-write
- disconnect-client
- ldif-export
- lockdown-mode
- manage-topology
- metrics-read
- modify-acl
- password-reset
- permit-get-password-policy-state-issues
- privilege-change
- server-restart
- server-shutdown
- soft-delete-read
- stream-values
- unindexed-search
- update-schema

The privileges not granted to root users by default includes:

- bypass-pw-policy
- bypass-read-acl
- jmx-read
- jmx-write
- jmx-notify
- permit-externally-processed-authentication
- permit-proxied-mschapv2-details
- proxied-auth

The set of default root privileges can be altered to add or remove values as necessary. Doing so will require the `config-read`, `config-write`, and `privilege-change` privileges, as well as either the `bypass-acl` privilege or sufficient permission granted by the access control configuration to make the change to the server's configuration.

Assigning additional privileges for administrators

To allow access to the Tasks backend, set up a global ACI that allows access to members of an Administrators group as follows:

```
$ dsconfig set-access-control-handler-prop \
  --add 'global-aci:(target="ldap:///cn=tasks")(targetattr="*||+")
        (version 5.0; acl "Access to the tasks backend for administrators";
         allow (all) groupdn="ldap:///
         cn=admins,ou=groups,dc=example,dc=com";)'
```

Assigning privileges to normal users and individual root users

Privileges can be granted to normal users on an individual basis. This can be accomplished by adding the `ds-privilege-name` operational attribute to that user's entry with the names of the desired privileges. For example, the following change will grant the `proxied-auth` privilege to the `uid=proxy,dc=example,dc=com` account:

```
dn: uid=proxy,dc=example,dc=com
changetype: modify
add: ds-privilege-name
ds-privilege-name: proxied-auth
```

The user making this change will be required to have the `privilege-change` privilege, and the server's access control configuration must also allow the requester to write to the `ds-privilege-name` attribute in the target user's entry.

This same method can be used to grant privileges to root users that they would not otherwise have through the set of default root privileges. You can also remove default root privileges from root users by prefixing the name of the privilege to remove with a minus sign. For example, the following change grants a root user the `jmx-read` privilege in addition to the set of default root privileges, and removes the `server-restart` and `server-shutdown` privileges:

```
dn: cn=Sync Root User,cn=Root DNs,cn=config
changetype: modify
add: ds-privilege-name
ds-privilege-name: jmx-read
ds-privilege-name: -server-restart
ds-privilege-name: -server-shutdown
```

Note that because root user entries exist in the configuration, this update requires the `config-read` and `config-write` privileges in addition to the `privilege-change` privilege.

Disabling privileges

Although the privilege subsystem in the PingDirectory Server is a very powerful feature, it might break some applications if they expect to perform some operation that requires a privilege that they do not have. In the vast majority of these cases, you can work around the problem by simply assigning the necessary privilege manually to the account used by that application. However, if this workaround is not sufficient, or if you need to remove a particular privilege (for example, to allow anyone to access information via JMX without requiring the `jmx-read` privilege), then privileges can be disabled on an individual basis.

The set of disabled privileges is controlled by the `disabled-privilege` property in the global configuration object. By default, no privileges are disabled. If a privilege is disabled, then the server behaves as if all users have that privilege.

Working with proxied authorization

The Directory Server supports the Proxied Authorization Control (RFC 4370) to allow an authorized LDAP client to authenticate to the server as another user. Typically, LDAP servers are deployed as backend authentication systems that store user credentials and authorization privileges necessary to carry out an operation. Single sign-on (SSO) systems can retrieve user credentials from the Directory Server and then issue permissions that allow the LDAP client to request operations under the identity as another user. The use of the proxied authorization control provides a means for client applications to securely process requests without the need to bind or re-authenticate to the server for each and every operation.

The Directory Server supports the proxied authorization V1 and V2 request controls. The proxied authorization V1 request control is based on early versions of the draft-weltman-ldapv3-proxy Internet draft and is available primarily for legacy systems. It is recommended that deployments use the proxied authorization V2 request control based on RFC 4370.

The proxied authorization V2 control is used to request that the associated operation be performed as if it has been requested by some other user. This control may be used in conjunction with add, delete, compare, extended, modify, modify DN, and search requests. In that case, the associated operation will be processed under the authority of the specified authorization identity rather than the identity associated with the client connection (i.e., the user as whom that connection is bound). The target authorization identity for this control is specified as an "authzid" value, which should be either "dn:" followed by the distinguished name of the target user, or "u:" followed by the user name.

Note that because of the inherent security risks associated with the use of the proxied authorization control, most directory servers that support its use enforce strict restrictions on the users that are allowed to request this control. If a user attempts to use the proxied authorization V2 request control and does not have sufficient permission to do so, then the server will return a failure response with the `AUTHORIZATION_DENIED` result code.

Configuring proxied authorization

About this task

Configuring proxied authorization requires a combination of access control instructions and the `proxied-auth` privilege to the entry that will perform operations as another user.

> ⓘ **Note:** You cannot use the `cn=Directory Manager` root DN as a proxying DN.

Steps

1. Open a text editor and create a user entry, such as `uid=clientApp`, which is the user entry that will request operations as another user, `uid=admin,dc=example,dc=com`. The client application entry also requires the `proxied-auth` privilege to allow it to run proxied authorization requests. Save the file as `add-user.ldif`.

```
dn: ou=Applications,dc=example,dc=com
objectClass: top
objectClass: organizationalUnit
objectClass: extensibleObject
ou: Admins
ou: Applications

dn: uid=clientApp,ou=Applications,dc=example,dc=com
objectClass: top
objectClass: person
objectClass: organizationalPerson
objectClass: inetOrgPerson
givenName: Client
uid: clientApp
cn: Client App
sn: App
userPassword: password
ds-privilege-name: proxied-auth
```

2. Add the file using **`ldapmodify`**.

```
$ bin/ldapmodify --defaultAdd --filename add-user.ldif
```

3. The client application targets a specific subtree in the Directory Information Tree (DIT) for its operations. For example, some client may need access to an accounts subtree to retrieve customer information. Another client may need access to another subtree, such as a subscriber subtree. In this example, we want the client application to target the `ou=People,dc=example,dc=com` subtree. To allow the target, open a text editor and create an LDIF file to assign an ACI to that branch so that the client app user can access it as a proxy auth user. Note that the ACI should be on a single line of text. The example shows the ACI over multiple lines for readability. Add the file using the **`ldapmodify`**.

```
dn: ou=People,dc=example,dc=com
changetype: modify
add: aci
aci:  (version 3.0; acl "People Proxy Access"; allow(proxy)
  userdn="ldap:///uid=clientApp,ou=Applications,dc=example,dc=com";)
```

4. Run a search to test the configuration using the bind DN `uid=clientApp` and the `proxyAs` option, which requires that you prefix "dn:" to the proxying entry or "u:" to the user name. The `uid=clientApp`

binds to the server and proxies as `uid=admin` to access the `ou=People,dc=example,dc=com` subtree.

```
$ bin/ldapsearch --port 1389 \
  --bindDN "uid=clientApp,ou=Applications,dc=example,dc=com" \
  --bindPassword password \
  --proxyAs "dn:uid=admin,dc=example,dc=com" \
  --baseDN ou=People,dc=example,dc=com \
  "(objectclass=*)"
```

Restricting proxy users

The Directory Server provides a set of operational attributes that restricts the proxied authorization capabilities of a client application and its proxyable target entry. When present in an entry, the Directory Server evaluates each operational attribute together to form a whitelist of potential users that can be proxied. If none of those attributes is present, then the user may potentially proxy as anyone.

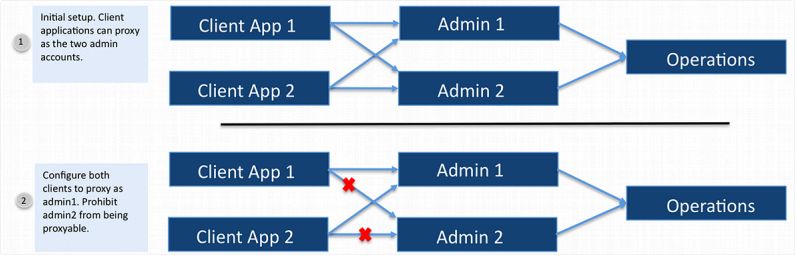
The Directory Server supports a two-tier provision system that, when configured, can restrict specific users for proxied authorization. The first tier is a set of `ds-auth-may-proxy-as-*` operational attributes on the client entry that will bind to the server and carry out operations under the identity of another user. The second tier is a set of `ds-auth-is-proxyable-*` operational attributes on the user entry that defines whether access is allowed, prohibited, or required by means of proxied authorization. If allowed or required, the attributes define which client entries can proxy as the user.

MY TITLE Proxying Operational Attributes

For example, if a client application, such as `uid=clientApp`, is requesting to search the `ou=People,dc=example,dc=com` branch as the user `uid=admin`, the command would look like this:

```
ldapsearch --bindDN uid=clientApp,dc=example,dc=com \
--bindPassword password \
--proxyAs uid=admin,dc=example,dc=com \
--baseDN ou=People,dc=example,dc=com \
"(object-class=*)
```

At bind, the Directory Server evaluates the list of users in the `uid=clientApp` entry based on the presence of any `ds-auth-may-proxy-as-*` attributes. In the figure below, the `uid=clientApp` entry has a `ds-auth-may-proxy-as` attribute with a value, `uid=admin`, which means that the client app user may proxy only as the `uid=admin` account. Next, the server confirms that `uid=admin` is in the list of proxyable users and then evaluates the `ds-auth-is-proxyable-*` attributes present in the `uid=admin` entry. These attributes determine the list of restricted users that either are allowed, prohibited, or required to proxy as the `uid=admin` entry. In this case, the `uid=admin` entry has the `ds-auth-is-proxyable` attribute with a value of "required", which indicates that the entry can only be accessed by means of proxied authorization. The `uid=admin` entry also has the `ds-auth-is-proxyable-by` attribute with a value of `uid=clientApp`, which indicates it can only be requested by the `uid=clientApp` entry. Once both sets of attributes have been confirmed, the `uid=clientApp` can bind to the server as the authenticated user. From this point, the Directory Server performs ACI evaluation on the branch to determine if the requested user has access rights to the branch. If the branch is accessible by the `uid=clientApp` entry, and then the search request is processed.



MY TITLE Proxying Operational Attributes Examples

About the ds-auth-may-proxy-as-* operational attributes

The Directory Server first evaluates the list of potential users that can be proxied for the authenticated user based on the presence of the `ds-auth-may-*` operational attributes in the entry. These operational attributes are multi-valued and are evaluated together if all are present in an entry:

- **ds-auth-may-proxy-as**. Specifies the user DNs that the associated user is allowed to proxy as. For instance, based on the previous example, you could specify in the `uid=clientApp` entry that it can proxy operations as `uid=admin` and `uid=agent1`.

```
dn: uid=clientApp,ou=Applications,dc=example,dc=com
objectClass: top
...
ds-privilege-name: proxied-auth
ds-auth-may-proxy-as: uid=admin,dc=example,dc=com
ds-auth-may-proxy-as: uid=agent1,ou=admins,dc=example,dc=com
```

- **ds-auth-may-proxy-as-group**. Specifies the group DNs and its group members that the associated user is allowed to proxy as. For instance, you could specify that the potential users that the `uid=clientApp` entry can proxy as are those members who are present in the group `cn=Agents,ou=Groups,dc=example,dc=com`. This attribute is multi-valued, so that more than one group can be specified. Nested static and dynamic groups are also supported.

```
dn: uid=clientApp,ou=Applications,dc=example,dc=com
objectClass: top
...
ds-privilege-name: proxied-auth
ds-auth-may-proxy-as-group: cn=Agents,ou=Groups,dc=example,dc=com
```

- **ds-auth-may-proxy-as-url**. Specifies the DNs that are returned based on the criteria defined in an LDAP URL that the associated user is allowed to proxy as. For instance, the attribute specifies that the client can proxy as those entries that match the criteria in the LDAP URL. This attribute is multi-valued, so that more than one LDAP URL can be specified.

```
dn: uid=clientApp,ou=Applications,dc=example,dc=com
objectClass: top
...
ds-privilege-name: proxied-auth
ds-auth-may-proxy-as-url: ldap:///ou=People,dc=example,dc=com??sub?
(l=austin)
```

About the ds-auth-is-proxyable-* operational attributes

After the Directory Server has evaluated the list of users that the authenticated user can proxy as, the server checks to see if the requested authorized user is in the list. If the requested authorized user is present in the list, then the server continues processing the proxable attributes in the entry. If the requested authorized user is not present in the list, the bind will fail.

The operational attributes on the proxying entry are as follows:

- **ds-auth-is-proxyable**. Specifies whether the entry is proxyable or not. Possible values are: "allowed" (operation may be proxied as this user), "prohibited" (operations may not be proxied as this

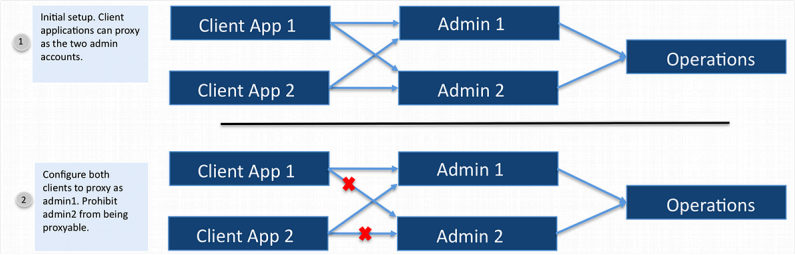
user), "required" (indicates that the account will not be allowed to authenticate directly but may only be accessed by some form of proxied authorization).

- **ds-auth-is-proxyable-as**. Specifies any users allowed to use this entry as a target of proxied authorization.
- **ds-auth-is-proxyable-as-group**. Specifies any groups allowed to use this entry as a target of proxied authorization. Nested static and dynamic groups are also supported.
- **ds-auth-is-proxyable-as-url**. Specifies the LDAP URLs that are used to determine any users that are allowed to use this entry as a target of proxied authorization.

Restricting proxied authorization for specific users

About this task

To illustrate how the proxied authorization operational attributes work, it is best to set up a simple example where two LDAP clients, `uid=clientApp1` and `uid=clientApp2` can freely proxy two administrator accounts, `uid=admin1` and `uid=admin2`. We will add the `ds-auth-may-proxy-as-*` and the `ds-auth-is-proxyable-*` attributes to these entries to restrict how each account can use proxied authorization. For example, the two client applications will continue to proxy the `uid=admin1` account but the `uid=admin2` account will no longer be able to be used as a proxied entry.



MY TITLE Proxy Users Example Scenario

Steps

1. For this example, set up two user entries, `uid=clientApp1` and `uid=clientApp2`, which will be proxying the `uid=admin1` and `uid=admin2` accounts to access the `ou=People,dc=example,dc=com` subtree. Both entries have the `proxied-auth` privilege assigned to it. Open a text editor and create an LDIF file. Add the file using the **`ldapmodify`** tool. Note that "..." indicates that other attributes present in the entry are not included in the example for readability purposes.

```
dn: uid=clientApp1,ou=Applications,dc=example,dc=com
objectClass: top
...
ds-privilege-name: proxied-auth

dn: uid=clientApp2,ou=Applications,dc=example,dc=com
objectClass: top
...
ds-privilege-name: proxied-auth
```

2. Next, assign the ACI for each client application to the subtree, `ou=People,dc=example,dc=com`. Note that the ACIs should be on one line of text. The example displays the ACIs over multiple lines for readability.

```
dn: ou=People,dc=example,dc=com
aci: (version 3.0; acl "People Proxy Access"; allow(proxy)
       userdn="ldap:///uid=clientApp1,ou=Applications,dc=example,dc=com";)
aci: (version 3.0; acl "People Proxy Access"; allow(proxy)
       userdn="ldap:///uid=clientApp2,ou=Applications,dc=example,dc=com";)
```

3. Run a search for each entry. In this example, assume that there are two admin accounts: `admin1` and `admin2` that have full access rights to user attributes. You should be able to proxy as the `uid=admin1` and `uid=admin2` entries to access the subtree for both clients.

```
$ bin/ldapsearch --port 1389 \
  --bindDN "uid=clientApp1,ou=Applications,dc=example,dc=com" \
  --bindPassword password \
  --proxyAs "dn:uid=admin1,dc=example,dc=com" \
  --baseDN ou=People,dc=example,dc=com \
  "(objectclass=*)"

$ bin/ldapsearch --port 1389 \
  --bindDN "uid=clientApp2,ou=Applications,dc=example,dc=com" \
  --bindPassword password \
  --proxyAs "dn:uid=admin2,dc=example,dc=com" \
  --baseDN ou=People,dc=example,dc=com \
  "(objectclass=*)"
```

4. Next, limit the proxied authorization capabilities for each client application. Update the `uid=clientApp1` entry to add the `ds-auth-may-proxy-as` attribute. In this example, the `ds-auth-may-proxy-as` attribute specifies that `uid=clientApp1` can proxy as the `uid=admin1` entry. Open a text editor, create the following LDIF file, save it, and add it using **`ldapmodify`**. Note that `ds-auth-may-proxy-as` is multi-valued:

```
dn: uid=clientApp1,ou=Applications,dc=example,dc=com
changetype: modify
add: ds-auth-may-proxy-as
ds-auth-may-proxy-as: uid=admin1,dc=example,dc=com
```

5. Repeat the previous step for the `uid=clientApp2` entry, except specify the `ds-auth-may-proxy-as-url`. The client entry may proxy as any DN that matches the LDAP URL.

```
dn: uid=clientApp2,ou=Applications,dc=example,dc=com
changetype: modify
add: ds-auth-may-proxy-as-url
ds-auth-may-proxy-as-url: ldap:///dc=example,dc=com??sub?(uid=admin*)
```

6. Next, we want to create a group of client applications that has `uid=clientApp1` and `uid=clientApp2` as its `uniquemembers` to illustrate the use of the `ds-auth-proxyable-by-group` attribute. In this example, set up a static group using the `groupOfUniqueNames` object class.

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

dn: cn=Client Applications,ou=Groups,dc=example,dc=com
objectClass: top
objectClass: groupOfUniqueNames
cn: Client Applications
ou: groups
uniquemember: uid=clientApp1,ou=Applications,dc=example,dc=com
uniquemember: uid=clientApp2,ou=Applications,dc=example,dc=com
```

7. Update the `uid=admin1` entry to provide the DN that it may be proxied as. Add the `ds-auth-is-proxyable` and the `ds-auth-is-proxyable-by` attributes. For instance, we make the `uid=admin1` a required proxyable entry, which means that it can only be accessed by some form of proxied authorization. Then, specify each DN that can proxy as `uid=admin1` using the `ds-auth-is-proxyable-by`. Open a text editor, create the following LDIF file, save it, and add it using **`ldapmodify`**. Note that the example includes all three types of `ds-auth-is-proxable-by-*`

attributes as an illustration, but, in an actual deployment, only one type of attribute is necessary if they all target the same entries.

```
dn: uid=admin1,dc=example,dc=com
changetype: modify
add: ds-auth-is-proxyable
ds-auth-is-proxyable: required
-
add: ds-auth-is-proxyable-by
ds-auth-is-proxyable-by: ou=clientApp1,ou=Applications,dc=example,dc=com
ds-auth-is-proxyable-by: ou=clientApp2,ou=Applications,dc=example,dc=com
-
add: ds-auth-is-proxyable-by-group
ds-auth-is-proxyable-by-group: cn=Client
 Applications,ou=Groups,dc=example,dc=com
-
add: ds-auth-is-proxyable-by-url
ds-auth-is-proxyable-by-url: ldap:///ou=Applications,dc=example,dc=com??sub?
(uid=clientApp*)
```

8. Next, prohibit proxying for the `uid=admin2` entry by setting the `ds-auth-is-proxyable` to prohibited. Open a text editor, create the following LDIF file, save it, and add it using **`ldapmodify`**.

```
dn: uid=admin2,dc=example,dc=com
changetype: modify
add: ds-auth-is-proxyable
ds-auth-is-proxyable: prohibited
```

9. Run a search using the proxied account. For example, run a search first with `uid=clientApp1` or `uid=clientApp2` that proxies as `uid=admin1` to return a successful operation. However, if you run a search for `uid=clientApp1` that proxies as `uid=admin2`, as seen below, you will see an "authorization denied" message due to `uid=admin2` not matching the list of potential entries that can be proxied. The `ds-auth-may-proxy-as-*` attributes specify that the client can only proxy as `uid=admin1`:

```
$ bin/ldapsearch --port 1389 \
  --bindDN "uid=clientApp1,ou=Applications,dc=example,dc=com" \
  --bindPassword password \
  --proxyAs "dn:uid=admin2,dc=example,dc=com" \
  --baseDN ou=People,dc=example,dc=com \
  "(objectclass=*)"
```

```
One of the operational attributes (ds-auth-may-proxy-as,
ds-auth-may-proxy-as-group, ds-auth-may-proxy-as-url) in user entry
'uid=clientApp1,ou=Applications,dc=example,dc=com' does not allow
that user to be proxied as user 'uid=admin2,dc=example,dc=com'

Result Code:  123 (Authorization Denied)

Diagnostic Message:  One of the operational attributes (ds-auth-may-proxy-
as,
ds-auth-may-proxy-as-group, ds-auth-may-proxy-as-url) in user entry
'uid=clientApp1,ou=Applications,dc=example,dc=com' does not allow that
user to be proxied as user 'uid=admin2,dc=example,dc=com'
```

10. Run another search using `uid=clientApp2`, which attempts to proxy as `uid=admin2`.
You will see an "authorization denied" message due to the presence of the `ds-auth-is-proxyable:prohibited` operational attribute, which states that `uid=admin2` is not available for proxied authorization.

```
$ bin/ldapsearch --port 1389 \
  --bindDN "uid=clientApp2,ou=Applications,dc=example,dc=com" \
  --bindPassword password \
```

```
  --proxyAs "dn:uid=admin2,dc=example,dc=com" \
  --baseDN ou=People,dc=example,dc=com \
  "(objectclass=*)"
```

```
The 'ds-auth-is-proxyable' operational attribute
in user entry 'uid=admin2,dc=example,dc=com' indicates that
user may not be accessed via proxied authorization

Result Code:  123 (Authorization Denied)

Diagnostic Message: The 'ds-auth-is-proxyable' operational
attribute in user entry 'uid=admin2,dc=example,dc=com' indicates
that user may not be accessed via proxied authorization
```

Working with parameterized ACIs

The Directory Server supports the use of parameterized ACIs to control access to subtrees with homogenous administrative group or user patterns, which can be used in multi-tenant deployments. A single parameterized ACI can take the place of specifying identical ACIs on each tenant's subtree. For example, the following parameterized ACI:

```
(target="ldap:///o=($1),dc=example,dc=com")(version 3.0; acl \
"Subtree Admin Group members may search for and read entries in their
 subtree."; allow \
(search, read) groupdn="ldap:///cn=Subtree Admin
 Group,ou=groups,o=($1),dc=example,dc=com";)
```

Enables:

- Members of a group with DN **"cn=Subtree Admin Group,ou=groups,o=Customers, dc=example,dc=com"** to search for and read entries in the **"o=Customers, dc=example,dc=com"** subtree.
- Members of a group with DN **"cn=Subtree Admin Group,ou=groups,o=Partners, dc=example,dc=com"** to search for and read entries in the **"o=Partners, dc=example,dc=com"** subtree

The same access is granted for any substitution value for the **($1)** parameter variable. If an operation tried to read the **uid=user.1,o=acme,dc=example,dc=com** entry, this ACI would be considered. This ACI would allow a read action, if the operation's user is a member of the **cn=Subtree Admin Group,ou=groups,o=acme,dc=example,dc=com** group.

Attribute values from the target DN can be replaced with different variables **($#)** and then reference those variables in the group DN or user DN. The string representation of a parameter variable is constructed as follows:

- an open parenthesis
- a dollar sign
- a positive integer
- a closing parenthesis

In another example:

```
"population=($2),ou=Populations,environment=($1),ou=Environments,o=Acme"
```

The **($2)** variable is the population ID in the DN of the target entry, and **($1)** is the environment ID in the DN of the target entry. Those values from the target entry's DN are then substituted into the group DN or user DN value.

Parameter variables present in a parameterized ACI's target will be associated with the actual values from the resource DN. Each actual value will be substituted for its respective parameter variable in the ACI's target, and group bind rule DNs when performing access control on the resource entry. Parameter

variables can be used in multiple RDNs in a parameterized target. A given RDN may have at most one parameter variable as its attribute value, and a given parameter variable may appear only once in the parameterized target.

The following values are examples of valid parameterized target DNs:

- **`ou=($1),dc=example,dc=com`**
- **`population=($2),ou=Populations,environment=($1),ou=Environments,o=Acme`**
- **`o=($1)`** (for a global ACI)

An ACI on an entry can only apply to that entry's subtree. If an ACI with a parameterized target is stored on an entry, that entry's DN must appear in a non-parameterized form as the rightmost RDNs of the parameterized target's DN. For example, if an ACI with a parameterized target were stored on the **`dc=example,dc=com`** entry, that parameterized target must end in **`dc=example,dc=com`** in a non-parameterized form. Global ACIs do not have this restriction. Each global ACI can have parameter variables in any or all of its parameterized target's RDNs. Additional restrictions for parameterized targets include:

- They may not be pattern ACIs. That is, they may not contain wildcards ('*').
- RDNs that are parameterized must be single-valued. For example, a given parameterized RDN may not consist of two or more type-value pairs joined by '+'.

$attr.attrName macro

The ($attr.attrName) macro extracts a value from a specified attribute in the target entry rather than extracting a value from a field with the same number in the target DN. For example, ($attr.description) is replaced with the value of the description attribute in the target entry. If there are multiple values for the specified attribute then multiple actualized DNs are produced for the bind DN and the first matching actualized DN is used.

The ($attr.attrName) macro expands only the attribute's value so that you can extract values from the targe DN and build RDNs with different attribute names and types. Since the ($attr.attrName) macro extracts only the attribute's value, it is possible to combine it with a type that is different than what is present in the target DN.

Managing the Schema

This chapter presents a basic summary of the supported schema components on the PingDirectory Server and procedures to extend the schema with new element definitions. The chapter presents the following topics:

About the Schema

A schema is the set of directory server rules that define the structures, contents, and constraints of a Directory Information Tree (DIT). The schema guarantees that any new data entries or modifications meet and conform to these predetermined set of definitions. It also reduces redundant data definitions and provides a uniform method for clients or applications to access its Directory Server objects.

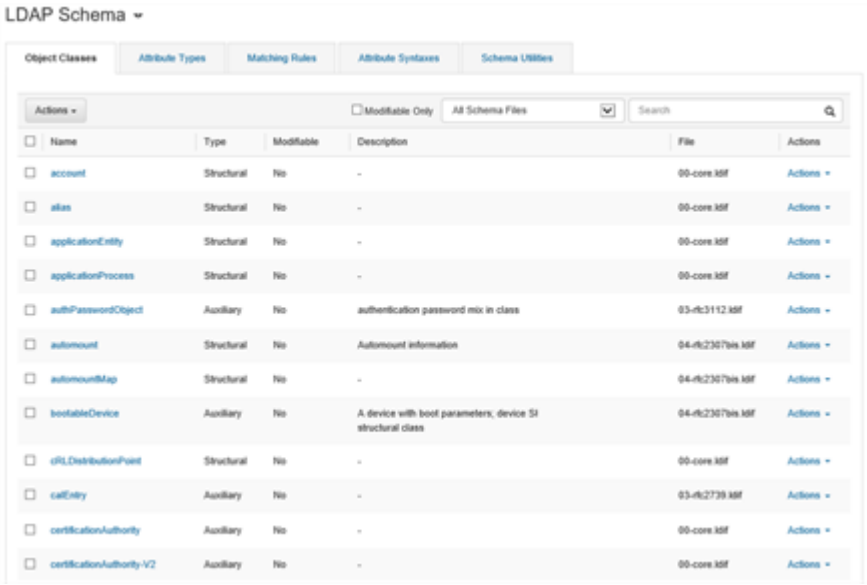
The PingDirectory Server ships with a default set of read-only schema files that define the core properties for the Directory Server. The Administrative Console provides a Schema Editor that administrators can use to view existing schema definitions and add new custom schema elements to their DIT. Any attempt to alter a schema element defined in a read-only file or add a new schema element to a read-only file will result in an "Unwilling to Perform" result.

About the Schema Editor

The Administrative Console provides a user-friendly graphical editor with tabs to manage any existing schema component related to the DIT: object classes, attributes, matching rules, attribute syntaxes, and

schema utilities. The **Object Classes** and **Attribute Types** tabs enable viewing existing definitions as well as adding, modifying, or removing custom schema elements.

The **Matching Rules** and **Attribute Syntaxes** tabs are read-only and provide a comprehensive listing of all of the elements necessary to define new schema elements. The **Schema Utilities** tab provides a schema validator that allows you to load a schema file or perform a cut-and-paste operation on the schema definition to verify that it meets the proper schema and ASN.1 formatting rules. The **Utilities** tab also supports schema file imports by first checking for proper syntax compliance and generating any error message if the definitions do not meet specification.



MY TITLE Example Schema Editor Screen

The Schema Editor provides two views for each definition: **Properties View** and **LDIF View**. The **Properties View** breaks down the schema definition by its properties and shows any inheritance relationships among the attributes. The **LDIF View** shows the equivalent schema definition in ASN.1 format, which includes the proper text spacing required for each schema element.

Default Directory Server Schema Files

The PingDirectory Server stores its schema as a set of LDIF files for a Directory Server instance in the `<server-root>/config/schema` directory. The Directory Server reads the schema files in alphanumeric order at startup, so that the `00-core.ldif` file is read first, then `01-pwpolicy.ldif`, and then the rest of the files. Custom schema files should be named so that they are loaded in last. For example, custom schema elements could be saved in a file labelled `99-user.ldif`, which loads after the default schema files are read at startup.

The Directory Server then uses the schema definitions to determine any violations that may occur during add, modify, or import requests. Clients applications check the schema (i.e., matching rule definitions) to determine the assertion value algorithm used in comparison or search operations.

The default set of schema files are present at installation and should not be modified. Modifying the default schema files could result in an inoperable server.

The schema files have the following descriptions:

Default Schema Files

Schema Files	Description
00-core.ldif	Governs the Directory Server's core functions.

Schema Files	Description
01-pwpolicy.ldif	Governs password policies.
02-config.ldif	Governs the Directory Server's configuration.
03-changelog.ldif	Governs the Directory Server's change log.
03-rfc2713.ldif	Governs Java objects.
03-rfc2714.ldif	Governs Common Object Request Broker Architecture (CORBA) object references.
03-rfc2739.ldif	Governs calendar attributes for vCard.
03-rfc2926.ldif	Governs Server Location Protocol (SLP) mappings to and from LDAP schemas.
03-rfc2985.ldif	Governs PKCS #9 public-key cryptography.
03-rfc3112.ldif	Governs LDAP authentication passwords.
03-rfc3712.ldif	Governs printer services.
03-uddiv3.ldif	Governs web services registries of SOA components.
04-rfc2307bis.ldif	Governs mapping entities from TCP/IP and UNIX into X.500 entries.

Extending the Directory Server Schema

The PingDirectory Server stores its schema as LDIF files in the `<server-root>/config/schema` directory. At startup, the Directory Server reads the schema files once in alphanumeric order starting with `00-core.ldif` and ending with any custom schema definition files, such as `99-user.ldif` if present.

You can extend the schema to include additional customizations necessary for your Directory Server data using one of the following methods:

- **Using the Schema Editor**. This method is the easiest and quickest way to set up a schema definition and have it validated for the correct ASN.1 formatting. The Editor lets you define your schema properties, load your custom file, or perform a cut-and-paste operation on a new schema element. If any errors exist in the file, the Schema Editor generates an error message if the schema definitions do not pass compliance.
- **Using a Custom Schema File**. You can create a custom schema file with your new definitions using a text editor, save it as `99-user.ldif`, and then import the file using the Schema Editor or the **`ldapmodify`** tool. You must name the custom LDIF file with a high two-digit number prefix, so that the Directory Server will read the file AFTER the core schema files are read at startup. For example, you can name the file, `99-myschema.ldif`, etc. See the next section, *General Tips on Extending the Schema* to see the requirements for naming each file.
- **Using the Command Line**. If you have a small number of additions, you can extend the schema over LDAP and from the command line using the **`ldapmodify`** tool. The Directory Server writes the new schema changes to a file `99-user.ldif` in the `<server-root>/config/schema` directory. However, this method can be cumbersome as schema definitions require strict adherence to text spacing and white space characters.

General Tips on Extending the Schema

You should consider the following points when extending the schema:

- Never modify the default schema files as doing so could damage the Directory Server’s processing capabilities.
- Define all attributes first before they can be used in an object class. If you are using the Schema Editor to add new schema elements, then you can use the Quick Add Attributes option when defining new objectclasses.

- Define the parent object class first before creating object classes that inherit from the parent.
- The schema file naming syntax requires that custom schema files must begin with exactly two digits followed by a non-digit character, followed by a zero or more characters and ending with ".ldif". Note that the two digits do not need to be followed by a dash ("-"). Any files that do not meet this criteria will be ignored and either a NOTICE or SEVERE_WARNING message will be logged.

  Any file in the `<server-root>/config/schema` directory with a name that starts with "." or with a name that ends with a tilde ('~'), ".swp", or ".tmp" will generate a NOTICE message indicating that temporary files will be ignored. Any other file that does not meet the naming criteria will generate a SEVERE_WARNING message indicating that it will be ignored.
- Define custom attributes and object classes in one file. Typically, this file will be the `99-user.ldif`. You can specify a different file name to which the Directory Server writes using the X-SCHEMA-FILE element and the file name in the definition. For example:

```
add: attributeTypes attributeTypes: ( 1.3.6.1.4.1.32473.3.1.9.1
  NAME 'contractorStatus'
  EQUALITY booleanMatch
  SYNTAX 1.3.6.1.4.1.1466.115.121.1.7
  SINGLE-VALUE
  USAGE userApplications
  X-ORIGIN 'Directory Server Example'
  X-SCHEMA-FILE '99-custom.ldif' )
```

- Pay special attention to the white space characters in the schema definitions, where WSP means zero or more space characters, and SP means one or more space characters. The LDIF specification states that LDIF parsers should ignore exactly one space at the beginning of each continuation line, since continuation lines must begin with a space character. Thus, if you define a new schema definition with each keyword on a separate continuation line, you should add two spaces before an element keyword to be safe. For example, the following attribute definition has two spaces before the keywords: NAME, SUP, and X-ORIGIN.

```
attributeTypes: ( 2.5.4.32 NAME 'owner' SUP distinguishedName X-ORIGIN 'RFC
 4519' )
```

- In a replicated topology, any new schema additions will be replicated to other replication servers to their respective Schema backend. The additions will be written to the file specified by the X-SCHEMA-FILE extension or written to `99-user.ldif` if no file is specified.

Managing Attribute Types

An attribute type determines the important properties related to an attribute, such as specifying the matching and syntax rules used in value comparisons. An attribute description consists of an attribute type and a set of zero or more options. Options are short, case-insensitive text strings that differentiate between attribute descriptions. For example, the LDAPv3 specification defines only one type of option, the tagging option, which can be used to tag language options, such as `cn;lang-de;lang-sp` or binary data, such as `userCertificate;binary`. You can also extend the schema by adding your own attribute definitions.

Attributes have the following properties:

- Attributes can be user attributes that hold information for client applications, or operational attributes that are used for administrative or server-related purposes. You can specify the purpose of the attribute by the `USAGE` element.
- Attributes are multi-valued by default. Multi-valued means that attributes can contain more than one value within an entry. Include the `SINGLE-VALUE` element if the attribute should contain at most one value within an entry.
- Attributes can inherit properties from a parent attribute as long as they both have the same USAGE, and the child attribute has the same `SYNTAX` or its `SYNTAX` allows values which are a subset of the values allowed by the `SYNTAX` of the parent attribute. For example, the surname (`sn`) attribute is a child of the name attribute.

Attribute Type Definitions

New attribute types do not require server code extensions if the provided matching rules and attribute syntaxes are used in the definitions. Administrators can create new attributes using the Schema Editor, which stores the definition in a file in the `<server-root>/config/schema` directory. See *Extending the Directory Server Schema* for more information.

The formal specification for attribute types is provided in RFC 4512, section 4.1.2 as follows:

```
AttributeTypeDescription = "(" wsp; Left parentheses followed by a white space
numericoid                      ; Required numeric object identifier
[ sp "NAME" sp qdescrs ]        ; Short name descriptor as alias for the OID
[ sp "DESC" sp qdstring ]       ; Optional descriptive string
[ sp "OBSOLETE" ]               ; Determines if the element is active
[ sp "SUP" sp oid ]             ; Specifies the supertype
[ sp "EQUALITY" sp oid ]        ; Specifies the equality matching rule
[ sp "ORDERING" sp oid ]        ; Specifies ordering matching rule
[ sp "SUBSTR" sp oid ]          ; Specifies substrings matching rule
[ sp "SYNTAX" sp oidlen ]       ; Numeric attribute syntax with minimum
 upper bound
                                ;  length expressed in {num}
[ sp "SINGLE-VALUE" ]           ; Specifies if the attribute is single
 valued in
                                ;  the entry
[ sp "COLLECTIVE" ]             ; Specifies if it is a collective attribute
[ sp "NO-USER-MODIFICATION" ]   ; Not modifiable by external clients
[ sp "USAGE" sp usage ]         ; Application usage
extensions wsp ")"              ; Extensions followed by a white space and
 ")"

usage = "userApplications" /    ; Stores user data
  "directoryOperation" /        ; Stores internal server data
  "distributedOperation" /      ; Stores operational data that must be
 synchronized
                                ; across servers
  "dSAOperation"                ; Stores operational data specific to a
 server and
                                ; should not be synchronized across servers
```

The following extensions are specific to the PingDirectory Server and are not defined in RFC 4512:

```
extensions = /
"X-ORIGIN" /                ; Specifies where the attribute type is defined
"X-SCHEMA-FILE" /           ; Specifies which schema file contains the definition
"X-APPROX" /                ; Specifies the approximate matching rule
"X-ALLOWED-VALUE" /         ; Explicitly specifies the set of allowed values
"X-VALUE-REGEX" /           ; Specifies the set of regular expressions to compare
 against
                            ;  attribute values to determine acceptance
"X-MIN-VALUE-LENGTH" /      ; Specifies the minimum character length for
 attribute values
"X-MAX-VALUE-LENGTH" /      ; Specifies the maximum character length for
 attribute values
"X-MIN-INT-VALUE" /         ; Specifies the minimum integer value for the
 attribute
"X-MAX-INT-VALUE" /         ; Specifies the maximum integer value for the
 attribute
"X-MIN-VALUE-COUNT" /       ; Specifies the minimum number of allowable values
 for the
                            ;  attribute
"X-MAX-VALUE-COUNT" /       ; Specifies the maximum number of allowable values
 for the
                            ;  attribute
```

```
"X-READ-ONLY"             ; True or False. Specifies if the file that contains
 the
                          ;  schema element is marked as read-only in the
 server
                          ;  configuration.
```

Basic Properties of Attributes

The Basic Properties section displays the standard elements in schema definition.

Basic Properties of Attributes

Attributes	Description
Name	Specifies the globally unique name.
Description	Specifies an optional definition that describes the attribute and its contents. The analogous LDIF equivalent is "DESC".
OID	Specifies the object identifier assigned to the schema definition. You can obtain a specific OID for your company that allows you to define your own object classes and attributes from IANA or ANSI.
Syntax	Specifies the attribute syntax used. For example, the `userPassword` attribute uses the User Password Syntax whereas the `authPassword` attribute uses the Authentication Password Syntax.
Parent	Specifies the schema definition's parent or supertype if any. The analogous LDIF equivalent is "SUP".
Multivalued	Specifies if the attribute can appear more than once in its containing object class.
Required By Class	Specifies any object classes that require the attribute.
Allowed By Class	Specifies any object classes that can optionally use the attribute.
Value Restrictions	Specifies any restriction on the value of the attribute.

The Extra Properties section provides additional auxiliary information associated with the attribute.

Basic Properties of Attributes

Attributes	Description
Aliases	Specifies any shortform alias names, if any. In theory, you could have any number of shortform names as long as they are all unique. The analogous LDIF equivalent appears as the secondary element with the NAME element. For example, NAME ('sn' 'surname').
Origin	Specifies the origin of the schema definition. Typically, it could refer to a specific RFC or company.
Stored in File	Specifies the schema file that stores the definition in the `<server-root>/config/schema` folder.
Usage	Specifies the intended use of the attribute. Choices are the following: ▪ userApplications ▪ directoryOperation ▪ distributedOperation ▪ dSAOperation
User-Modifiable	Specifies if the attribute can be modified by an authorized user.

Attributes	Description
Obsolete	Specifies if the schema definition is obsolete or not.
Matching Rules	Specifies the associated matching rules for the attribute.

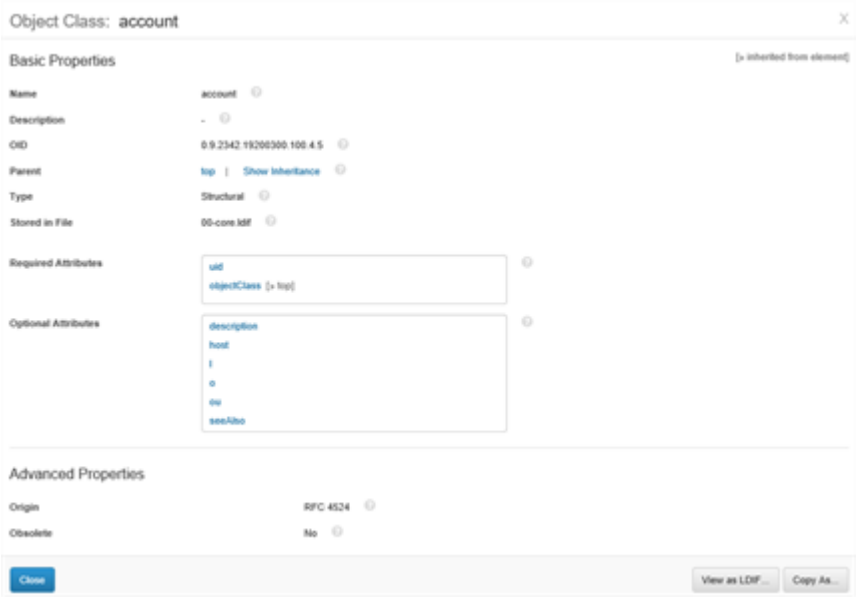
Viewing Attributes

The Schema Editor displays all of the attribute types on your directory server instance. It shows the basic properties that are required elements plus the extra properties that are allowed within the attribute definition.

To View Attribute Types Using the Schema Editor

Steps

1. Start the Administrative Console. Check that the Directory Server instance associated with the console is also running.
2. On the main menu, click **Schema**.
3. In the Administrative Console **Schema Editor**, click the **Attribute Types** tab.
4. Click a specific attribute to view its definition. In this example, click the `account` attribute. In the **Object Class** window, view the attribute properties.



5. Click the **View as LDIF** button to see the equivalent attribute definition in ASN.1 format.

To View Attribute Types over LDAP

Steps

- Use **`ldapsearch`** to view a multi-valued operational attribute `attributeTypes`, which publishes the definitions on the Directory Server. The attribute is stored in the subschema subentry.

```
$ bin/ldapsearch --baseDN cn=schema --searchScope base \
  "(objectclass=*)" attributeTypes
```

To View a Specific Attribute Type over LDAP

Steps

- Use **ldapsearch** with the `--dontWrap` option and use the **grep** command to search for a specific attribute.

```
$ bin/ldapsearch --baseDN cn=schema \
  --searchScope base --dontWrap "(objectclass=*)" \
  attributeTypes | grep 'personalTitle'
```

Creating a New Attribute over LDAP

The following section shows how you can add the schema element from the previous section over LDAP. You can create your own schema file or type the schema from the command line. In either case, you must pay special attention to text spacing and ASN.1 formatting.

To Add an New Attribute to the Schema over LDAP

Steps

1. Create an LDIF file with the new attribute definition using a text editor. Save the file as `myschema.ldif`.

```
dn: cn=schema
changetype: modify
add: attributeTypes
attributeTypes: ( contractorStatus-OID NAME 'contractorStatus'
  EQUALITY booleanMatch
  SYNTAX 1.3.6.1.4.1.1466.115.121.1.7
  SINGLE-VALUE
  USAGE userApplications
  X-ALLOWED-VALUES 'Y' 'N' 'y' 'n'
  X-ORIGIN 'PingDirectory Server Example' )
```

2. Use **ldapmodify** to add the attribute.

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

3. Verify the addition by displaying the attribute using **ldapsearch**.

```
$ bin/ldapsearch --baseDN cn=schema --searchScope base \
  --dontwrap "(objectclass=*)" attributeTypes | grep 'contractorStatus'
```

4. You can view the custom schema file at `<server-root>/config/schema/99-user.ldif`. You should see the following:

```
dn: cn=schema
objectClass: top
objectClass: ldapSubentry
objectClass: subschema
cn: schema
attributeTypes: ( contractorStatus-OID
  NAME 'contractorStatus'
  EQUALITY booleanMatch
  SYNTAX 1.3.6.1.4.1.1466.115.121.1.7
  SINGLE-VALUE
  USAGE userApplications
  X-ORIGIN 'PingDirectory Server Example' )
```

To Add Constraints to Attribute Types

Steps

- The Directory Server provides attribute type extensions that constrain the values for the associated attribute using the `DirectoryString` attribute syntax. The following schema definition includes two `attributeType` definitions for `myAttr1` and `myAttr2`. The first definition constrains the values for the attribute `myAttr1` to 'foo', 'bar', 'baz'. The second definition constrains the minimum allowable length for `myAttr2` to 1 and the maximum allowable length to 5.

```
attributeTypes: (1.2.3.4
  NAME 'myAttr1'
  SYNTAX 1.3.6.1.4.1.1466.115.121.1.15
  X-ALLOWED-VALUES ( 'foo' 'bar' 'baz' ))
attributeTypes: ( 1.2.3.5
  NAME 'myAttr2'
  SYNTAX 1.3.6.1.4.1.1466.115.121.1.15
  X-MIN-VALUE-LENGTH '1'
  X-MAX-VALUE-LENGTH '5' )
```

Managing Object Classes

Object classes are sets of related information objects that form entries in a Directory Information Tree (DIT). The Directory Server uses the schema to define these entries, to specify the position of the entries in a DIT, and to control the operation of the server. You can also extend the schema by adding your own schema definitions.

Object classes have the following general properties:

- Object classes must have a globally unique name or identifier.
- Object classes specify the required and allowed attributes in an entry.
- Object classes can inherit the properties and the set of allowed attributes from its parent object classes, which may also be part of a hierarchical chain derived from the top abstract object class.
- Object classes that are defined in the PingDirectory Server can be searched using the `objectClasses` operational attribute. The Directory Server also has a special entry called the subschema subentry, which provides information about the available schema elements on the server.

Object Classes Types

Based on RFC 4512, object classes can be a combination of three different types:

- **Abstract object classes** are used as the base object class, from which structural or auxiliary classes inherit its properties. This inheritance is a one-way relationship as abstract object classes cannot be derived from structural or auxiliary classes. The most common abstract object class is `top`, which defines the highest level object class in a hierarchical chain of object classes.
- **Structural object classes** define the basic attributes in an entry and define where an entry can be placed in a DIT. All entries in a DIT belong to one structural object class. Structural object classes can inherit properties from other structural object classes and from abstract object classes to form a chain of inherited classes. For example, the `inetOrgPerson` structural object class inherits properties from the `organizationalPerson` structural class, which inherits from another object class, person.
- **Auxiliary object classes** are used together with structural object classes to define additional sets of attributes required in an entry. The auxiliary object class cannot form an entry alone but must be present with a structural object class. Auxiliary object classes cannot derive from structural object classes or vice-versa. They can inherit properties from other auxiliary classes and from abstract classes.

Object Class Definition

New object classes can be specified with existing schema components and do not require additional server code extensions for their implementation. Administrators can create new object classes using the Schema

Editor, which manages schema in the `<server-root>/config/schema` directory. See *Extending the Directory Server Schema* for more information.

The object class definition is defined in RFC 4512, section 4.1.1, as follows::

```
ObjectClassDescription = "(" wsp; Left parenthesis followed by a white space
numericoid                      ; Required numeric object identifier
[ sp "NAME" sp qdescrs ]        ; Short name descriptor as alias for the OID
[ sp "DESC" sp qdstring ]       ; Optional descriptive string
[ sp "OBSOLETE" ]               ; Determines if the element is inactive
[ sp "SUP" sp oid ]             ; Specifies the direct superior object class
[ sp kind ]                     ; abstract, structural (default), auxiliary
[ sp "MUST" sp oids ]           ; Required attribute types
[ sp "MAY" sp oids ]            ; Allowed attribute type
extensions wsp ")"              ; Extensions followed by a white space and
 ")"

usage = "userApplications" /    ; Stores user data
  "directoryOperation" /        ; Stores internal server data
  "distributedOperation" /      ; Stores operational data that must be
 synchronized
                                ;   across servers
  "dSAOperation"                ; Stores operational data specific to a
 server and
                                ;   should not be synchronized across
 servers
```

The following extensions are specific to the PingDirectory Server and are not defined in RFC 4512:

```
extensions = /
"X-ORIGIN" /             ; Specifies where the object class is defined
"X-SCHEMA-FILE" /        ; Specifies which schema file contains the definition
"X-READ-ONLY"            ; True or False. Specifies if the file that contains
                         ;   the schema element is marked as read-only
                         ;   in the server configuration.
```

ⓘ **Note:** Although RFC 4512 allows multiple superior object classes, the PingDirectory Server allows at most one superior object class, which is defined by the SUP element in the definition.

Basic Object Class Properties

The Basic Properties section displays the standard elements in schema definition.

Basic Properties of Attributes

Attributes	Description
Name	Specifies the globally unique name.
Description	Specifies an optional definition that describes the object class and its contents. The analogous LDIF equivalent is "DESC".
OID	Specifies the object identifier assigned to the schema definition. You can obtain a specific OID for your company that allows you to define your own object classes and attributes from IANA or ANSI.
Parent	Specifies the schema definition's hierarchical parent or superior object class if any. An object class can have one parent. The analogous LDIF equivalent
Type	Specifies the type of schema definition: abstract, structural, or auxiliary. The analogous LDIF equivalent is "ABSTRACT", "STRUCTURAL", or "AUX

Attributes	Description
Required Attributes	Specifies any required attributes with the object class. The Schema Editor also marks any inherited attributes from another object class. You can double-click an attribute value to take you to the Properties View for that particular attribute. The analogous LDIF equivalent is "MUST".
Optional Attributes	Specifies any optional attributes that could be used with the object class. The Schema Editor also marks any inherited attributes from another object class. You can double-click an attribute value to take you to the Properties View for that particular attribute. The analogous LDIF equivalent is "MAY".

The Extra Properties section provides additional auxiliary information associated with the object class.

Basic Properties of Attributes

Attributes	Description
Aliases	Specifies any shortform alias names, if any. In theory, you could have any number of shortform names as long as they are all unique. The analogous LDIF equivalent appears as the secondary element with the NAME element although most object classes do not have aliases.
Origin	Specifies the origin of the schema definition. Typically, it could refer to a specific RFC or company.
Obsolete	Specifies if the schema definition is obsolete or not.
Stored in File	Specifies the schema file that stores the definition in the `<server-root>/config/schema` folder.

Viewing Object Classes

You can view the object classes on your Directory Server by using the Administrative Console Schema Editor, over LDAP using the **`ldapsearch`** tool, or some third party tool. The Schema Editor displays all of the object classes on the directory server instance. It shows the basic properties that are required elements and the extra properties that are allowed within the object class.

To View Object Classes over LDAP

Steps

- Use **`ldapsearch`** tool to view a multi-valued operational attribute, `objectClasses`, which publishes the object class definitions on the Directory Server. The attribute is stored in the subschema subentry.

```
$ bin/ldapsearch --baseDN cn=schema --searchScope base \
--dontWrap "(objectclass=*)" objectClasses
```

```
dn: cn=schema
objectClasses: ( 2.5.6.0 NAME 'top' ABSTRACT MUST objectClass X-ORIGIN 'RFC
 4512' )
objectClasses: ( 2.5.6.1 NAME 'alias' SUP top STRUCTURAL MUST
 aliasedObjectName
  X-ORIGIN 'RFC 4512' )
objectClasses: ( 2.5.6.2 NAME 'country' SUP top STRUCTURAL MUST c
  MAY ( searchGuide $ description ) X-ORIGIN 'RFC 4519' )
...(more output)...
```

Managing an Object Class over LDAP

The following section shows how you can manage an object class schema element over LDAP by adding a new attribute element to an existing object class. You can create your own schema file or type the schema from the command line. In either case, you must pay special attention to text spacing and ASN.1 formatting. The following example procedure adds an attribute, `contractorAddress`, to the custom schema file, then adds it to the `contractor` objectclass.

To Manage an Object Class over LDAP

Steps

1. Assume that you have defined the `contractorAddress` attribute, create an LDIF file called `contractorAddress-attr.ldif` with the following content:

```
dn: cn=schema
changetype: modify

add: attributeTypes attributeTypes: ( contractor-OID NAME
 'contractorAddress'
  SYNTAX 1.3.6.1.4.1.1466.115.121.1.15
  SINGLE-VALUE
  USAGE userApplications
  X-ORIGIN 'user defined'
  X-SCHEMA-FILE '98-custom-schema.ldif' )
  X-ORIGINS 'user defined'
  X-SCHEMA-FILE '98-custom-schema.ldif' )
```

2. Add the attribute using **`ldapmodify`**.

```
$ bin/ldapmodify --filename contractorAddress-attr.ldif
```

3. Next, create an LDIF file to modify the contractor objectclass to allow this attribute. When doing this, you are just re-adding the updated `objectClass` and the Directory Server will handle the proper replacement of the existing object class with the new one. Create a file called `contractor-oc.ldif`. Make sure that the lines are not wrapped, the `objectClasses` line should be one continuous line.

```
dn:cn=schema
changetype: modify
add: objectClasses
objectClasses: ( contractor-OID NAME 'contractor'
  DESC 'Contractor status information
  SUP top
  AUXILIARY MAY ( contractorStatus $ contractorAgency $ contractorAddress )
  X-ORIGIN 'Directory Server Example'
  X-SCHEMA-FILE '98-custom-schema.ldif' )
```

4. Update the `objectClass` using **`ldapmodify`** as follows:

```
$ bin/ldapmodify --filename contractor-oc.ldif
```

5. These schema changes will be replicated to all servers in the replication topology. Verify the change by looking at the `config/schema/98-custom-schema.ldif` file on the other servers in the replication topology to ensure that the changes are present.
6. If you need to add an index for this attribute, you can do so by using the **`dsconfig`** command-line utility. You will need to do this on each server in your topology unless you have server configuration groups set up. See *Configuring Server Groups* for more information.

```
$ bin/dsconfig create-local-db-index --backend-name userRoot \
  --index-name contractorAddress --set index-type:equality
```

7. Rebuild the index online. This will not affect other indexes or entries since there is no currently existing data for this attribute on any entry.

```
$ bin/rebuild-index --baseDN dc=example,dc=com --index contractorAddress
```

Creating a New Object Class Using the Schema Editor

The procedures to create a new object class are similar to that of creating a new attribute. Make sure that any attributes that are part of the new object class are defined prior to defining the object class.

To Create a New Object Class Using the Schema Editor

Steps

1. Start the Administrative Console.
2. On the main menu, click **LDAP Schema**.
3. On the **Schema Editor**, click the **Object Classes** tab, and then click **New** located in the bottom left of the window.
4. Enter the properties for the new objectclass. In the **Attributes** box, filter the types of attributes required for the new object class. Click the right arrow to move it into the **Required** or the **Optional** box. All custom attributes appear at the bottom of the list in the **Attributes** box.

Extending the Schema Using a Custom Schema File

You can add new attributes and object classes to your Directory Server schema by creating a custom schema file. You can import the file using the Schema Editor, over LDAP using the **`ldapmodify`** tool, or from the command line. Make sure to define the attributes first, then define the object classes.

To Extend the Schema Using a Custom Schema File

Steps

1. Create an LDIF file with the new attribute extensions using a text editor.

```
dn: cn=schema
objectClass: top
objectClass: ldapSubentry
objectClass: subschema
attributeTypes: ( contractorStatus-OID NAME 'contractorStatus'
  EQUALITY booleanMatch
  SYNTAX 1.3.6.1.4.1.1466.115.121.1.7
  SINGLE-VALUE
  USAGE userApplications
  X-ORIGIN 'Directory Server Example' )
attributeTypes: ( contractorAgency-OID NAME 'contractorAgency'
  EQUALITY caseIgnoreMatch
  SUBSTR caseIgnoreSubstringsMatch
  SYNTAX 1.3.6.1.4.1.1466.115.121.1.44{256}
  SINGLE-VALUE
  USAGE userApplications
  X-ORIGIN 'PingDirectory Server Example' )
```

2. In the same LDIF file, add a new object class definition after the attribute types. In this example, create an auxiliary object class, `contractor`, that alone cannot be used as an entry. The object class will be used to add supplemental information to the `inetOrgPerson` structural object class. The attributes are all optional for the new object class.

```
objectClasses: ( contractor-OID
  NAME 'contractor'
  DESC 'Contractor status information'
```

```
  SUP top
  AUXILIARY
  MAY ( contractorStatus $ contractorAgency )
  X-ORIGIN 'PingDirectory Server Example' )
```

3. Save the file as `99-auxobjclass.ldif`. and place it in the `<server-root>/config/schema` directory.
4. At this stage, the schema extensions are not loaded into the Directory Server yet. You have four options to load them:
   - Create a task that loads the new extensions into the schema. We create a task labelled with the ID "add-schema-99-auxobjclass" and add it using **`ldapmodify`**. The server does not need to be restarted using this method.

     ```
     dn: ds-task-id=add-schema-99-auxobjclass,cn=Scheduled Tasks,cn=tasks
     objectClass: top
     objectClass: ds-task
     objectClass: ds-task-add-schema-file
     ds-task-id: add-schema-99-auxobjclass
     ds-task-class-name:
      com.unboundid.directory.server.tasks.AddSchemaFileTask
     ds-task-schema-file-name: 99-auxobjclass.ldif
     ```
   - Import the schema file using the Administrative Console Schema Editor. You do not need to restart the server when using this method.
   - Place the `99-auxobjclass.ldif` file in the `<server-root>/config/schema` directory and restart the Directory Server. The schema file is read at startup.
   - Add the schema file using **`load-ldap-schema-file`**. You do not need to restart the server when using this method.

     ```
     $ bin/load-ldap-schema-file --schemaFile config/schema 99-auxobjclass.ldif
     ```
5. Verify the addition by displaying the attribute using **`ldapsearch`**.

   ```
   $ bin/ldapsearch --baseDN dc=example,dc=com "(uid=user.9)" contractorStatus
   ```

   ```
   dn: uid=user.9,ou=People,dc=example,dc=com
   contractorStatus: TRUE
   ```

Managing Matching Rules

Matching rules determine how clients and servers compare attribute values during LDAP requests or operations. They are also used in evaluating search filter elements including distinguished names and attributes. Matching rules are defined for each attribute based on EQUALITY (e.g., two attributes are equal based on case, exact match, etc.), SUBSTR (e.g., assertion value is a substring of an attribute), and ORDERING (e.g., greater than or equal, less than or equal, etc.) properties.

ⓘ **Note:** The PingDirectory Server supports an `APPROXIMATE` matching rule that compares similar attributes based on fuzzy logic. Thus, attributes that are similar or "sound-like" each other are matched. For example, "petersen" would match "peterson".

Matching Rule Definition

New matching rules require additional server code extensions to be implemented on the PingDirectory Server. If you need new matching rules, contact your authorized support provider for assistance.

The formal specification for attribute types is provided in RFC 4512, section 4.1.3 as follows:

```
MatchingRuleDescription = "(" wsp    ; Left parentheses followed by a white
 space
numericoid                          ; Required numeric object identifier
 identifying
```

```
                                        ;   this matching rule
[ sp "NAME" sp qdescrs                  ; Short name descriptor
[ sp "DESC" sp qdstring                 ; Description
[ sp "OBSOLETE" ]                       ; Specifies if the rule is inactive
sp "SYNTAX" sp numericoid               ; Assertion syntax
extensions wsp ")"                      ; Extensions followed by a white space
 and ")"
```

Default Matching Rules

The PingDirectory Server provides a large set of matching rules, which support a variety of applications. The default matching rules available for the Directory Server are listed in the table below for each matching rule type: Equality, Substring, Ordering, and Approximate matches.

Default Matching Rules

Matching Rule/OID	Attribute Syntax/OID	Description
uuidMatch/ 1.3.6.1.1.16.2	UUID/ 1.3.6.1.1.16.1	Compares an asserted UUID with a stored UUID attribute value for equality. RFC 4530.
uuidOrderingMatch/ 1.3.6.1.1.16.3	UUID/ 1.3.6.1.1.16.1	Compares the collation order of an asserted UUID with a stored UUID attribute value for ordering RFC 4530.
caseExactIA5Match/ 1.3.6.1.4.1.1466.109.114.1	IA5 String/ 1.3.6.1.4.1.1466.115.121.1.26	Compares an asserted value with an attribute value of International Alphabet 5 syntax. RFC 4517.
caseIgnoreIA5Match/ 1.3.6.1.4.1.1466.109.114.2	IA5 String/ 1.3.6.1.4.1.1466.115.121.1.26	Compares an asserted value with an attribute value of International Alphabet 5 syntax. Case, leading, and trailing spaces are ignored. Multiple spaces are treated as a single space. RFC 4517.
caseIgnoreIA5SubstringsMatch/ 1.3.6.1.4.1.1466.109.114.3	Substring Assertion/ 1.3.6.1.4.1.1466.115.121.1.58	Compares an asserted substring with an attribute value of IA5 string syntax. Case, leading, and trailing spaces are ignored. Multiple spaces are treated as a single space. RFC 4517.
authPasswordExactMatch/ 1.3.6.1.4.1.4203.1.2.2	Authentication Password Syntax/ 1.3.6.1.4.1.4203.1.1.2	Authentication password exact matching rule.
authPasswordMatch/ 1.3.6.1.4.1.4203.1.2.3	Authentication Password Syntax/ 1.3.6.1.4.1.4203.1.1.2	Authentication password matching rule.
ds-mr-double-metaphone-approx/ 1.3.6.1.4.1.30221.1.4.1	Directory String/ 1.3.6.1.4.1.1466.115.121.1.15	Syntax based on the phonetic Double Metaphone algorithm for approximate matching.
ds-mr-user-password-exact/ 1.3.6.1.4.1.30221.1.4.2	User Password Syntax/ 1.3.6.1.4.1.30221.1.3.1	User password exact matching rule.

Matching Rule/OID	Attribute Syntax/OID	Description
ds-mr-user-password-equality/ 1.3.6.1.4.1.30221.1.4.3	User Password Syntax/ 1.3.6.1.4.1.30221.1.3.1	User password equality matching rule.
historicalCsnOrderingMatch/ 1.3.6.1.4.1.30221.1.4.4	1.3.6.1.4.1.30221.1.3.5	Compares the collation order of a historical change sequence number with a historical CSN attribute value.
caseExactIA5SubstringsMatch/ 1.3.6.1.4.1.30221.1.4.902	Substring Assertion/ 1.3.6.1.4.1.1466.115.121.1.58	Compares an asserted substring with an attribute value of IA5 string syntax. RFC 4517.
compactTimestampMatch/ 1.3.6.1.4.1.30221.2.4.1	Compact Timestamp/ 1.3.6.1.4.1.30221.2.3.1	Compact Timestamp matching rule.
compactTimestampOrderingMatch/ 1.3.6.1.4.1.30221.2.4.2	Compact Timestamp/ 1.3.6.1.4.1.30221.2.3.1	Compares the collation order of a compact timestamp number with an attribute value of Compact Timestamp syntax.
objectIdentifierMatch/ 2.5.13.0	OID/ 1.3.6.1.4.1.1466.115.121.1.38	Compares an asserted value with an attribute value of OID syntax. RFC 4517.
distinguishedNameMatch/ 2.5.13.1	DN/ 1.3.6.1.4.1.1466.115.121.1.12	Compares an asserted value with an attribute value of DN syntax. Spaces around commas and semicolons are ignored. Spaces around plus and equal signs around RDN components are ignored. RFC 4517.
caseIgnoreMatch/ 2.5.13.2	Directory String/ 1.3.6.1.4.1.1466.115.121.1.15	Compares an asserted value with an attribute value. Case, leading, and trailing spaces are ignored. Multiple spaces are treated as a single space. RFC 4517.
caseIgnoreOrderingMatch/ 2.5.13.3	Directory String/ 1.3.6.1.4.1.1466.115.121.1.15	Compares the collation order of the asserted string with an attribute value of Directory String syntax. Case, leading, and trailing spaces are ignored. Multiple spaces are treated as a single space. RFC 4517

Matching Rule/OID	Attribute Syntax/OID	Description
caseIgnoreSubstringsMatch/ 2.5.13.4	Substring Assertion/ 1.3.6.1.4.1.1466.115.121.1.58	Compares an asserted substring value with an attribute value of Directory String syntax. Case, leading, and trailing spaces are ignored. Multiple spaces are treated as a single space. RFC 4517
caseExactMatch/ 2.5.13.5	Directory String/ 1.3.6.1.4.1.1466.115.121.1.15	Compares an asserted value with an attribute value of Directory String syntax. RFC 4517.
caseExactOrderingMatch/ 2.5.13.6	Directory String 1.3.6.1.4.1.1466.115.121.1.15	Compares the collation order of the asserted string with an attribute value of Directory String syntax. RFC 3698
caseExactSubstringsMatch/ 2.5.13.7	Substring Assertion/ 1.3.6.1.4.1.1466.115.121.1.58	Compares an asserted substring with an attribute value of Directory String syntax. RFC 3698
numericStringMatch/ 2.5.13.8	Numeric String/ 1.3.6.1.4.1.1466.115.121.1.36	Compares an asserted value with an attribute value of Numeric String syntax. Spaces are ignored when performing these comparisons. RFC 4517.
numericStringOrderingMatch/ 2.5.13.9	Numeric String/ 1.3.6.1.4.1.1466.115.121.1.36	Compares the collation order of the asserted string with an attribute value of Numeric String syntax. Spaces are ignored when performing these comparisons. RFC 4517.
numericStringSubstringsMatch/ 2.5.13.10	Substring Assertion/ 1.3.6.1.4.1.1466.115.121.1.58	Compares an asserted substring with an attribute value of Numeric String syntax. Spaces are ignored when performing these comparisons. RFC 4517.
caseIgnoreListMatch/ 2.5.13.11	Postal Address/ 1.3.6.1.4.1.1466.115.121.1.41	Compares an asserted value with an attribute value which is a sequence of Directory Strings. Case, leading, and trailing spaces are ignored. Multiple spaces are treated as a single space. RFC 4517.

Matching Rule/OID	Attribute Syntax/OID	Description
caseIgnoreListSubstringsMatch/ 2.5.13.12	Substring Assertion/ 1.3.6.1.4.1.1466.115.121.1.58	Compares the asserted substring with an attribute value, which is a sequence of Directory Strings. Case, leading, and trailing spaces are ignored. Multiple spaces are treated as a single space. RFC 3698.
booleanMatch/ 2.5.13.13	Boolean/ 1.3.6.1.4.1.1466.115.121.1.7	Compares an asserted boolean value with an attribute value of BOOLEAN syntax. Returns true if the values are both TRUE or both FALSE. RFC 3698.
integerMatch/ 2.5.13.14	Integer/ 1.3.6.1.4.1.1466.115.121.1.27	Compares an asserted value with an attribute value of INTEGER syntax. RFC 4517.
integerOrderingMatch/ 2.5.13.15	Integer/ 1.3.6.1.4.1.1466.115.121.1.27	Compares the collation order of the asserted integer with an attribute value of Integer syntax. Returns true if the attribute value is less than the asserted value. RFC 3698.
bitStringMatch/ 2.5.13.16	Bit String/ 1.3.6.1.4.1.1466.115.121.1.6	Compares an asserted Bit String value with an attribute value of Bit String syntax. RFC 4517.
octetStringMatch/ 2.5.13.17	Octet String/ 1.3.6.1.4.1.1466.115.121.1.40	Compares an asserted value with an attribute value of octet string syntax using a byte-for-byte comparison. RFC 4517.
octetStringOrderingMatch/ 2.5.13.18	Octet String/ 1.3.6.1.4.1.1466.115.121.1.40	Compares the collation order of the asserted octet string with an attribute value of Octet String syntax. Zero precedes a one bit. Shorter strings precede longer strings. RFC 3698.
octetStringSubstringsMatch/ 2.5.13.19	Substring Assertion/ 1.3.6.1.4.1.1466.115.121.1.58	Compares an asserted substring with an attribute value of octet string syntax using a byte-for-byte comparison. RFC 4517.
telephoneNumberMatch/ 2.5.13.20	Telephone Number/ 1.3.6.1.4.1.1466.115.121.1.50	Compares an asserted value with an attribute value of Telephone Number syntax. RFC 4517.

Matching Rule/OID	Attribute Syntax/OID	Description
telephoneNumberSubstringsMatch/ 2.5.13.21	Substring Assertion/ 1.3.6.1.4.1.1466.115.121.1.58	Compares an asserted value with the substrings of an attribute value of Telephone Number String syntax. RFC 4517.
presentationAddressMatch/ 2.5.13.22	Presentation Address/ 1.3.6.1.4.1.1466.115.121.1.43	Compares an asserted value with an attribute value of Presentation Address syntax. RFC 4517.
uniqueMemberMatch/ 2.5.13.23	Name and Optional UID/ 1.3.6.1.4.1.1466.115.121.1.34	Compares an asserted value with an attribute value of Unique Member syntax. RFC 4517.
protocolInformationMatch/ 2.5.13.24	Protocol Information/ 1.3.6.1.4.1.1466.115.121.1.42	Compares an asserted value with an attribute value of Protocol Information syntax. RFC 4517.
generalizedTimeMatch/ 2.5.13.27	Generalized Time/ 1.3.6.1.4.1.1466.115.121.1.24	Compares an asserted value with an attribute value of Generalized Time syntax. RFC 4517.
generalizedTimeOrderingMatch/ 2.5.13.28	Generalized Time 1.3.6.1.4.1.1466.115.121.1.24	Compares the collation order of the asserted string with an attribute value of Generalized Time String syntax and case is ignored. RFC 4517.
integerFirstComponentMatch/ 2.5.13.29	Integer/ 1.3.6.1.4.1.1466.115.121.1.27	Equality matching rules for subschema attributes between an Integer syntax and the value syntax. RFC 4517.
objectIdentifierFirstComponentMatch/ 2.5.13.30	OID/ 1.3.6.1.4.1.1466.115.121.1.38	Equality matching rules for subschema attributes between an OID syntax and the value syntax. RFC 4517.
directoryStringFirstComponentMatch/ 2.5.13.31	Directory String/ 1.3.6.1.4.1.1466.115.121.1.15	Compares an asserted Directory String value with an attribute value of type SEQUENCE whose first component is mandatory and of type Directory String. Returns true if the attribute value has a first component whose value matches the asserted Directory String using the rules of caseIgnoreMatch. RFC 3698.

Matching Rule/OID	Attribute Syntax/OID	Description
wordMatch/ 2.5.13.32	Directory String/ 1.3.6.1.4.1.1466.115.121.1.15	Compares an asserted word with any word in the attribute value for equality RFC 3698.
keywordMatch/ 2.5.13.33	Directory String/ 1.3.6.1.4.1.1466.115.121.1.15	Compares an asserted value with any keyword in the attribute value for equality. RFC 3698.

Basic Matching Rule Properties

The Properties section displays the standard elements in a matching rule schema definition.

Basic Properties of Matching Rules

Attributes	Description
Name	Specifies the descriptive and unique name of the element.
Description	Specifies an optional definition that describes the matching rule. The analogous LDIF equivalent is "DESC".
OID	Specifies the globally unique object identifier assigned to the schema definition. You can obtain a specific OID for your company that allows you to define your own object classes and attributes from IANA or ANSI.
Type	Specifies the type of type of matching rule: Equality, Ordering, Substring, or Approximate.
Syntax	Specifies the matching rule syntax.
Used by Attributes	Specifies any attributes that use the corresponding matching rule.

Viewing Matching Rules

You can view the matching rules on your Directory Server by using the Schema Editor on the Administrative Console, over LDAP using the **`ldapsearch`** tool, or some third party tool.

The Schema Editor displays all of the matching rules on your Directory Server instance. It shows the basic properties that are allowed within the matching rule.

To View Matching Rules Over LDAP

Steps

- Use **`ldapsearch`** to view a multi-valued operational attribute, `matchingRules`, which publishes the definitions on the Directory Server. The attribute is stored in the subschema subentry.

```
$ bin/ldapsearch --baseDN cn=schema --searchScope base \
  "(objectclass=*)" matchingRules
```

Managing Attribute Syntaxes

The attribute type definition has a `SYNTAX` element, or attribute syntax, that specifies how the data values for the attribute are represented. The syntax can be used to define a large range of data types necessary for client applications. An attribute syntax uses the Abstract Syntax Notation One (ASN.1) format for its definitions.

Attribute Syntax Definition

New attribute syntaxes require additional code to be implemented on the PingDirectory Server. If you need new syntax definitions, contact your authorized support provider for assistance.

The formal specification for attribute types is provided in RFC 4512, section 4.1.5 as follows:

```
SyntaxDescription = "(" wsp
numericoid                ; Object identifier
[ sp "DESC" sp qdstring ] ; Description
extensions wsp ")"        ; Extensions followed by a white space and ")"
```

Default Attribute Syntaxes

The PingDirectory Server supports a large set of Attribute Syntax rules for applications. The default Attribute Syntax rules available for the directory server are listed in the table below.

Default Attribute Syntaxes

LDAP Syntax	OID	Description
UUID	1.3.6.1.1.16.1	128-bit (16 octets) Universally Unique Identifier (UUID) used for Uniform Resource Names as defined in RFC 4122. For example, a4028c1a-f36e-11da-ba1a-04112154bd1e.
Attribute Type Description	1.3.6.1.4.1.1466.115.121.1.3	Syntax for the AttributeTypeDescription rule based on RFC 4517.
Binary	1.3.6.1.4.1.1466.115.121.1.5	Strings based on Basic Encoding Rules (BER) or Distinguished Encoding rules (DER). For example, an X.509 digital certificate or LDAP messages are BER encoded.
Bit String	1.3.6.1.4.1.1466.115.121.1.6	Sequence of binary digits based on RFC 4517. For example, '0010111'B.
Boolean	1.3.6.1.4.1.1466.115.121.1.7	TRUE or FALSE.
Certificate	1.3.6.1.4.1.1466.115.121.1.8	BER/DER-encoded octet strings based on an X.509 public key certificate as defined in RFC 4523.
Certificate List	1.3.6.1.4.1.1466.115.121.1.9	BER/DER-encoded octet string based on an X.509 certificate revocation list as defined in RFC 4523.
Certificate Pair	1.3.6.1.4.1.1466.115.121.1.10	BER/DER-encoded octet string based on an X.509 public key certificate pair as defined in RFC 4523.
Country String	1.3.6.1.4.1.1466.115.121.1.11	Two character country code specified in ISO 3166. For example, US, CA, etc.
DN	1.3.6.1.4.1.1466.115.121.1.12	Distinguished name of an entry as defined in RFC4514.
Delivery Method	1.3.6.1.4.1.1466.115.121.1.14	Sequence of services in preference order by which an entity receives messages as defined in RFC4517. For example, videotext $ telephone.

LDAP Syntax	OID	Description
Directory String	1.3.6.1.4.1.1466.115.121.1.15	String of one or more characters from the Universal Character Set (UCS) using UCS Transformation Format 8 (UTF-8) encoding of the string.
DIT Content Rule Description	1.3.6.1.4.1.1466.115.121.1.16	DITContentRuleDescription as defined in RFC4517.
DIT Structure Rule Description	1.3.6.1.4.1.1466.115.121.1.17	DITStructureRuleDesciption as defined in RFC4517.
Enhanced Guide	1.3.6.1.4.1.1466.115.121.1.21	Combination of attribute types and filter operators to be used to construct search filters as defined in RFC4517. For example, person#(sn$EQ)#oneLevel.
Facsimile Telephone Number	1.3.6.1.4.1.1466.115.121.1.22	Fax telephone number on the public switched telephone network as defined in RFC4517.
Fax	1.3.6.1.4.1.1466.115.121.1.23	Image generated using Group 3 fax process as defined in RFC4517.
Generalized Time	1.3.6.1.4.1.1466.115.121.1.24	String representing data and time as defined in RFC4517. YYYYMMDDHHMMSS[.|,fraction][(+|-HHMM)|Z] For example, 201103061032, 201103061032-0500, or 201103061032Z ("Z" indicates Coordinated Universal Time).
Guide	1.3.6.1.4.1.1466.115.121.1.25	Attribute types and filter operators as defined in RFC4517.
IA5 String	1.3.6.1.4.1.1466.115.121.1.26	String of zero or more characters from the International Alphabet 5 (IA5) character set as defined in RFC4517.
Integer	1.3.6.1.4.1.1466.115.121.1.27	String representations of integer values. For example, the character string "1234" represents the number 1234 as defined in RFC4517.
JPEG	1.3.6.1.4.1.1466.115.121.1.28	Image in JPEG File Interchange Format (JFIF) as defined in RFC4517.
Matching Rule Description	1.3.6.1.4.1.1466.115.121.1.30	MatchingRuleDescription as defined in RFC4512.
Matching Rule Use Description	1.3.6.1.4.1.1466.115.121.1.31	Attribute types to which a matching rule is applied in an extensibleMatch search filter (RFC4511).
Name and Optional UID	1.3.6.1.4.1.1466.115.121.1.34	Distinguished name and an optional unique identifier that differentiates identical DNs as defined in RFC4517. For example, uid=jsmith,ou=People,dc=example,dc=com#'0111'B
Name Form Description	1.3.6.1.4.1.1466.115.121.1.35	NameFormDescription as defined in RFC4512.

LDAP Syntax	OID	Description
Numeric String	1.3.6.1.4.1.1466.115.121.1.36	Sequence of one or more numerals and spaces as defined in RFC4517. For example, 14 848 929 102.
Object Class Description	1.3.6.1.4.1.1466.115.121.1.37	ObjectClassDescription as defined in RFC 4512.
OID	1.3.6.1.4.1.1466.115.121.1.38	Object identifier as defined in RFC 4512.
Other Mailbox	1.3.6.1.4.1.1466.115.121.1.39	Specifies an electronic mailbox as defined in RFC 4517. For example, otherMailbox = google $ user@gmail.com
Octet String	1.3.6.1.4.1.1466.115.121.1.40	Sequence of zero or more octets (8-bit bytes) as defined in RFC4517.
Postal Address	1.3.6.1.4.1.1466.115.121.1.41	Strings of characters that form a multi-line address in a physical mail system. Each component is separated by a "$". For example, 1234 Main St.$Austin, TX 78744$USA.
Protocol Information	1.3.6.1.4.1.1466.115.121.1.42	Undefined.
Presentation Address	1.3.6.1.4.1.1466.115.121.1.43	String encoded OSI presentation address as defined in RFC 1278. For example, TELEX +00728722+RFC-1006+03+10.0.0.6
Printable String	1.3.6.1.4.1.1466.115.121.1.44	String of one or more printable ASCII alphabetic, numeric, and punctuation characters as defined in RFC 4517.
RFC3672 Subtree Specification	1.3.6.1.4.1.1466.115.121.1.45	Syntax based on subtree specification as defined as RFC 3672.
Supported Algorithm	1.3.6.1.4.1.1466.115.121.1.49	Octet string based on the LDAP-encoding for a supported algorithm value that results from the BER encoding of a SupportedAlgorithm ASN.1 value.
Telephone Number	1.3.6.1.4.1.1466.115.121.1.50	String of printable international telephone number representations in E.123 format as defined in RFC 4517. For example, +1 512 904 5525.
Teletex Terminal Identifier	1.3.6.1.4.1.1466.115.121.1.51	Identifier and telex terminal as defined in RFC 4517.
Telex Number	1.3.6.1.4.1.1466.115.121.1.52	String representing the telex number, country code, and answerback code as defined in RFC 4517. For example, 812374, ch, ehhg.
UTC Time	1.3.6.1.4.1.1466.115.121.1.53	Character string representing the data and time in UTC Time format as defined as RFC 4517: YYMMDDHHMM[SS][(+|-HHMM)|Z], where Z is the coordinated universal time. For example, 0903051035Z, 0903051035-0500.
LDAP Syntax Description	1.3.6.1.4.1.1466.115.121.1.54	SyntaxDescription as defined in RFC4512.

LDAP Syntax	OID	Description
Substring Assertion	1.3.6.1.4.1.1466.115.121.1.58	Syntax for assertion values in an extensible match as defined in RFC 4517.
Authentication Password Syntax	1.3.6.1.4.1.4203.1.1.2	Encoded password storage syntax as defined in RFC 3112. For example, the syntax specifies the storage scheme in brackets as follows: <storage-scheme> $<auth component>$<auth value> For example: SSHA$xdEZWRqgyJk= $egDEFDXvdeeEnXUEIDPnd39dkpe=
User Password Syntax	1.3.6.1.4.1.30221.1.3.1	Encoded password storage syntax as defined in RFC 2307. For example, the syntax specifies the storage scheme in brackets as follows: {SSHA}XaljOF0ii3fOwCrU1klgBpWFayqSYs +5W1pMnw==
Relative Subtree Specification	1.3.6.1.4.1.30221.1.3.2	Similar to the RFC 3672 subtree specification except it uses an LDAP search filter as the specification filter.
Absolute Subtree Specification	1.3.6.1.4.1.30221.1.3.3	Syntax for a subset of entries in a subtree based on RFC 3672.
Sun-defined Access Control Information	1.3.6.1.4.1.30221.1.3.4	Syntax for access control instructions used in Sun Directory Servers.
Compact Timestamp	1.3.6.1.4.1.30221.2.3.1	Syntax based on compact timestamp ISO 8601 format. For example, 20110306T102532.

Basic Attribute Syntax Properties

The Properties section displays the standard elements in an attribute syntax.

Basic Properties of Attribute Syntaxes

Attributes	Description
Name	Specifies the descriptive and unique name of the element.
Description	Indicates an optional definition that describes the attribute syntax. The analogous LDIF equivalent is "DESC".
OID	Specifies the globally unique object identifier assigned to the schema definition.
Used by Attributes	Indicates any attributes that use the corresponding attribute syntax.

Viewing Attribute Syntaxes

You can view the attribute syntaxes on your Directory Server by using the Schema Editor on the Administrative Console, over LDAP using the **`ldapsearch`** tool, or some third party tool.

The Schema Editor displays all of the attribute syntaxes on your Directory Server instance. It shows the properties that are allowed within the attribute syntax.

To View Attribute Syntaxes Over LDAP

Steps

- Use **`ldapsearch`** to view the Directory Server's published list of attribute syntaxes using the multi-valued operational attribute, `ldapSyntaxes`, which publishes the definitions on the Directory Server. The attribute is stored in the subschema subentry.

```
$ bin/ldapsearch --baseDN cn=schema \
  --searchScope base "(objectclass=*)" ldapSyntaxes
```

Using the Schema Editor Utilities

The Schema Editor provides a **Schema Utilities** tab that enables importing new schema elements from a file and to checking schema compliance. If importing a schema file, the system automatically checks for compliance prior to the import. If the definition does not meet schema compliance, the system will display an error message. However, it is good practice to first check if your file is compliant with your schema prior to importing it.

To Check Schema Compliance Using the Schema Editor

Steps

1. Start the Administrative Console.
2. On the main menu, click **LDAP Schema**.
3. On the Schema Editor, click the **Schema Utilities** tab.
4. Click **Import Schema Elements** to read in an LDIF file, or copy-and-paste a new schema definition, and then click **Validate Entries**. If there is a problem, an error will be generated.

Modifying a Schema Definition

The Directory Server only allows schema definitions that are read-write to be edited. Schema elements indicated by the **Modifiable** column in the Schema Editor's tables can be modified.

To Modify a Schema Definition

Steps

1. Start the Administrative Console. Check that the Directory Server instance associated with the console is running.
2. On the main menu, click **Schema**.
3. On the Schema Editor, click the **Object Classes** tab.
4. Select the object class that you want to modify, and then click **Actions -> Edit**. The Edit dialog box appears.
5. Make your changes, and then click **OK**.

Deleting a Schema Definition

The Directory Server only allows schema definitions that are read-write to be deleted. In general, those schema definitions in the **Custom** folder of the Schema Editor can be removed from the system. You should make sure that the schema element is not currently in use.

To Delete a Schema Definition

Steps

1. Start the Administrative Console. Check that the Directory Server instance associated with the console is also running.
2. On the main menu, click **Schema**.
3. On the **Schema Editor**, click the **Object Classes** tab.
4. Select the object class that you want to remove, and then click **Actions -> Delete**.
5. On the **Confirmation** dialog, click **Yes** if you are sure that you want to delete the schema element.

Schema Checking

The PingDirectory Server provides full support for parsing all schema elements and provides access to all of its components. By default, the Directory Server enables schema checking for all operations, especially when importing data to the server or when modifying entries using the **`ldapmodify`** tool. Any schema violations will generate an error message to standard output.

To View the Schema Checking Properties

Steps

- Use **`dsconfig`** to view the schema checking property.

```
$ bin/dsconfig get-global-configuration-prop \
  --property check-schema
```

To Disable Schema Checking

About this task
Although not recommended, you can use the **`dsconfig`** tool to disable the schema checking. This feature only applies to public backends. Schema checking is enforced on private backends, such as changes to the Configuration, Schema, Task, and others. An admin action alert will be generated when attempting to disable schema checking using **`dsconfig`** interactive or non-interactive mode. The alert provides alternatives to disabling schema checking.

Steps

Run the **`dsconfig`** command and specify the `set-global-configuration-prop` subcommand to disable the `check-schema` property.

```
$ bin/dsconfig --no-prompt set-global-configuration-prop \
  --set check-schema:false
```

The system generates an admin action alert providing alternate options to disabling schema checking. Press **Enter** to continue the process or following one of the suggested tasks:

```
One or more configuration property changes require administrative action or
confirmation/notification.

Those properties include:

  *   check-schema: Schema checking should only be disabled as a last resort
      since disabling schema checking harms performance and can lead to
      unexpected behavior in the server as well as the applications that
      access it. There are less severe options for addressing schema issues:
```

```
 1.  Update the data to conform to the server schema.

 2.  Modify the server schema to conform to the data. Contact support before
     modifying the server's default schema.

 3.  Change the single-structural-objectclass-behavior property to allow
     entries to have no structural object class or multiple structural
object
     classes.

 4.  Change the invalid-attribute-syntax-behavior property to allow
attribute
     values to violate their attribute syntax.

 5.  Change the allow-zero-length-values property of the Directory String
     Attribute Syntax configuration to allow attributes with this syntax to
     have a zero length value.

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

Managing Matching Rule Uses

Matching Rule Use definitions map certain attribute types with a matching rule definition for extensible match filters. Extensible match filters allows clients to search using DN components, for example, (`ou:dn:=engineering`) or using an OID number, for example, (`cn:1.2.3.4:=Sam Carter`). The matching rule use attribute publishes those attribute types and matching rule combinations, which can be used in extensible match assertions.

Typically, you define a matching rule use that is not normally specified in the attribute type definition. You can create new matching rule uses from the existing schema definitions by adding a custom schema file in the `<server-root>/config/schema` directory.

Matching Rule Use Definitions

Matching Rule Use can be specified with existing schema components and do not require additional code for its implementation.

The formal specification for attribute types is provided in RFC 4512, section 4.1.4 as follows:

```
MatchingRuleUseDescription = "(" wsp
numericoid                  ; Object identifier
[ sp "NAME" sp qdescrs ]    ; Short name descriptor
[ sp "DESC" sp qdstring ]   ; Description
[ sp "OBSOLETE" ]           ; Specifies if the rule use is inactive
sp "APPLIES" sp oid         ; Attribute types
extensions wsp ")"          ; Extensions followed by a white space and ")"
```

The following extensions are specific to the PingDirectory Server and are not defined in RFC 4512:

```
extensions = /
"X-SCHEMA-FILE" /        ; Specifies which schema file contains the definition
"X-READ-ONLY"            ; True or False. Specifies if the file that contains
                         ;   the schema element is marked as read-only in the
                         ;   server configuration.
```

To View Matching Rule Uses

About this task

A matching rule use lists the attribute types that are suitable for use with an `extensibleMatch` search filter.

Steps

- Use **ldapsearch** to view the Directory Server's published list of matching rule uses using the multi-valued operational attribute, `matchingRuleUse`, which publishes the definitions on the Directory Server if any. The attribute is stored in the subschema subentry.

```
$ bin/ldapsearch --baseDN cn=schema --searchScope base \
  "(objectclass=*)" matchingRuleUse
```

Managing DIT Content Rules

DIT Content Rules provide a way to precisely define what attributes may be present in an entry, based on its structural object class, without specifically creating a new object class definition. The DIT content rules can define the mandatory and optional attributes that entries contain, the set of auxiliary object classes that entries may be part of, and any optional attributes from the structural and auxiliary object classes that are prohibited from being present in the entries.

DIT Content Rule Definitions

DIT Content Rules can be specified with existing schema components and do not require additional code for its implementation. On the PingDirectory Server, only one DIT Content Rule may be defined for an entry in the structural object class.

The formal specification for attribute types is provided in RFC 4512, section 4.1.6 as follows:

```
DITContentRuleDescription = "(" wsp
numericoid                  ; Object identifier of the structural object class
 the rule applies to
[ sp "NAME" sp qdescrs ]    ; Short name descriptor
[ sp "DESC" sp qdstring ]   ; Description
[ sp "OBSOLETE" ]           ; Specifies if the rule is inactive
[ sp "AUX" sp oids ]        ; List of allowed auxiliary object classes
[ sp "MUST" sp oids ]       ; List of required attributes
[ sp "MAY" sp oids ]        ; List of allowed attributes in the entry
[ sp "NOT" sp oids ]        ; List of prohibited attributes in the entry
extensions wsp ")"          ; Extensions followed by a white space and ")"
```

The following extensions are specific to the PingDirectory Server and are not defined in RFC 4512:

```
extensions = /
"X-ORIGIN" /             ; Specifies where the attribute type is defined
"X-SCHEMA-FILE" /        ; Specifies which schema file contains the definition
"X-READ-ONLY"            ; True or False. Specifies if the file that contains
                         ;   the schema element is marked as read-only in
                         ;   the server configuration.
```

To View DIT Content Rules

Steps

- Use **ldapsearch** to view a multi-valued operational attribute `dITContentRules`, which publishes the definitions on the Directory Server if any. The attribute is stored in the subschema subentry.

```
$ bin/ldapsearch --baseDN cn=schema --searchScope base \
  "(objectclass=*)" dITContentRules
```

Managing Name Forms

Name Forms define how entries can be named based on their structural object class. Specifically, name forms specify the structural object class to be named, as well as the mutually-exclusive set of required and allowed attributes to form the Relative Distinguished Names (RDNs) of the entries. Each structural object class may be associated with at most one name form definition.

Name Form Definitions

Name Forms can be specified with existing schema components and do not require additional code for its implementation.

The formal specification for attribute types is provided in RFC 4512, section 4.1.7.2 as follows:

```
NameFormDescription = "(" wsp
numericoid                  ; object identifier
[ sp "NAME" sp qdescrs ]    ; short name descriptor
[ sp "DESC" sp qdstring ]   ; description
[ sp "OBSOLETE" ]           ; not active
sp "OC" sp oid              ; structural object class
sp "MUST" SP oids           ; attribute types
[ sp "MAY" sp oids ]        ; attribute types
extensions wsp ")"          ; extensions followed by a white space and ")"
```

The following extensions are specific to the PingDirectory Server and are not defined in RFC 4512:

```
extensions = /
"X-ORIGIN" /             ; Specifies where the attribute type is defined
"X-SCHEMA-FILE" /        ; Specifies which schema file contains the definition
"X-READ-ONLY"            ; True or False. Specifies if the file that contains
                         ;   the schema element is marked as read-only in
                         ;   the server configuration.
```

To View Name Forms

Steps

- Use **`ldapsearch`** to view a multi-valued operational attribute `nameForms`, which publishes the definitions on the Directory Server. The attribute is stored in the subschema subentry.

  ```
  $ bin/ldapsearch --baseDN cn=schema --searchScope base "(objectclass=*)"
   nameForms
  ```

Managing DIT Structure Rules

DIT Structure Rules define which entries may be superior or subordinate to other entries in the DIT. Together with name forms, DIT Structure Rules determine how RDNs are added together to make up distinguished names (DNs). Because DITs do not have a global standard and are specific to a company's implementation, each DIT structure rule associates a name form with an object class and specifies each structure rule with an integer rule identifier, instead of an OID number. The identifier defines its relationship, either superior or subordinate, to another object class. If no superior rules are specified, then the DIT structure rule applies to the root of the subtree.

DIT Structure Rule Definition

DIT Structure Rules can be specified with existing schema components and do not require additional code for its implementation.

The formal specification for attribute types is provided in RFC 4512, section 4.1.7.1 as follows:

```
DITStructureRuleDescription = "(" wsp
ruleid                      ; object identifier
[ sp "NAME" sp qdescrs ]    ; short name descriptor
[ sp "DESC" sp qdstring ]   ; description
[ sp "OBSOLETE" ]           ; specifies if the rule is inactive
sp "FORM" sp oid            ; OID or name form with which the rule is
 associated
[ sp "SUP" ruleids ]        ; Superior rule IDs
extensions wsp ")"          ; extensions followed by a white space and ")"
```

The following extensions are specific to the PingDirectory Server and are not defined in RFC 4512:

```
extensions = /
"X-ORIGIN" /             ; Specifies where the rule is defined
"X-SCHEMA-FILE" /        ; Specifies which schema file contains the definition
"X-READ-ONLY"            ; True or False. Specifies if the file that contains
                         ;   the schema element is marked as read-only in
                         ;   the server configuration.
```

To View DIT Structure Rules

Steps

- Use **ldapsearch** to view a multi-valued operational attribute `dITStructureRules`, which publishes the definitions on the Directory Server. The attribute is stored in the subschema subentry.

```
$ bin/ldapsearch --baseDN cn=schema --searchScope base \
  "(objectclass=*)" dITStructureRules
```

Managing JSON Attribute Values

The PingDirectory Server supports a JSON object attribute syntax, which can be used for attribute types whose values are JSON objects. The syntax requires that each value of this type is a valid JSON object. The following is an example schema definition:

```
dn: cn=schema
objectClass: top
objectClass: ldapSubentry
objectClass: subschema
cn: schema
attributeTypes: ( jsonAttr1-OID NAME 'jsonAttr1' DESC 'test json attribute
 support' EQUALITY jsonObjectExactMatch SYNTAX 1.3.6.1.4.1.30221.2.3.4 USAGE
 userApplications )
objectClasses: ( jsonObjectClass-OID NAME 'jsonObjectClass' AUXILIARY MAY
 jsonAttr1 )
```

ⓘ **Note:** The EQUALITY matching rule should always be specified as `jsonObjectExactMatch` in the schema definition. Using the `jsonObjectFilterExtensibleMatch` is not valid in this case.

The `jsonObjectExactMatch` and `jsonObjectFilterExtensibleMatch` matching rules are provided to filter equality matching rule JSON object syntax. The following three additional matching rules are used in conjunction with the **jsonObjectExactMatch** and provide support for customizing the way that the server treats case sensitivity in JSON field names and in string values:

- **jsonObjectCaseSensitiveNamesCaseSensitiveValues**
- **jsonObjectCaseInsensitiveNamesCaseInsensitiveValues**
- **jsonObjectCaseInsensitiveNamesCaseSensitiveValues**

The `jsonObjectExactMatch` equality matching rule is used in evaluating equality filters in search operations, and for matching performed against JSON object attributes for add, compare, and modify operations. It determines whether two values are logically-equivalent JSON objects. The field names used in both objects must match exactly (although fields may appear in different orders). The values of each field must have the same data types. The order of elements in arrays is considered significant. Substring or approximate matching is not supported.

The `jsonObjectFilterExtensibleMatch` matching rule can perform more powerful matching against JSON objects. The assertion values for these extensible matching filters should be JSON objects that express the constraints for the matching. These JSON object filters are described in detail in the Javadoc documentation for the LDAP SDK for Java. Although the LDAP SDK can facilitate searches with this

matching rule, these searches can be issued through any LDAP client API that supports extensible matching.

The following are example searches using the `jsonObjectFilterExtensibleMatch` rule with available filter types.

Equals field filter type:

```
$ bin/ldapsearch -p 1389 -b dc=example,dc=com  -D "cn=Directory Manager"
 -w password '(jsonAttr1:jsonObjectFilterExtensibleMatch:={ "filterType" :
 "equals", "field" : ["stuff", "onetype", "name"], "value" : "John Doe" })'
```

Contains field filter type:

```
$ bin/ldapsearch -p 1389 -b dc=example,dc=com  -D "cn=Directory Manager"
 -w password '(jsonAttr1:jsonObjectFilterExtensibleMatch:={ "filterType" :
 "containsField", "field" : "age", "expectedType" : "number" })'
```

Greater than field filter type:

```
$ bin/ldapsearch -p 1389 -b dc=example,dc=com  -D "cn=Directory Manager"
 -w password '(jsonAttr1:jsonObjectFilterExtensibleMatch:={ "filterType" :
 "greaterThan", "field" : "age", "value" : 26, "allowEquals" : true})'
```

Configuring JSON Attribute Constraints

The PingDirectory Server can define a number of constraints for the fields included in JSON objects stored in values of a specified attribute type. Constraints that can be placed on a JSON field include:

- Requiring values of the field to have a specified data type.
- Indicating whether the field is required or optional.
- Indicating whether the field can have multiple values in an array. If a field is permitted to have array values, restrictions can also be placed on the number of elements that can be present in the array.
- Indicating whether the field can have a value that is the null primitive as an alternative to values of the indicated data type.
- Restricting values of string fields to a predefined set of values, that match a given regular expression, or a specified length.
- Restricting values of numeric fields with upper and lower bounds.

Any existing data that doesn't match newly-defined JSON constraints can still be decoded and managed by the server. Only new entries are subject to the new constraints. Attempts to alter existing entries with non-compliant JSON objects may require fixing those objects to make them conform to the new constraints.

The two global configuration properties that define schema constraints for JSON objects are `create-json-attribute-constraints` and `create-json-field-constraints` in `dsconfig`. In `dsconfig` interactive under advanced settings, the menu options are JSON Attribute Constraints and JSON Field Constraints. Configuration properties for each include:

- **attribute-type**. The name or object identifier of the attribute type with which the definition is associated. This attribute type must have the JSON object syntax. This property will be the naming attribute for the configuration entry.
- **allow-unnamed-fields**. A boolean value that indicates whether JSON objects, used as the values of attributes of the associated type, can include fields that are not referenced in the `attribute-value-constraints` object. If this is `false`, JSON objects will only be permitted to have the defined fields. If this is `true` (which is the default behavior), JSON objects are permitted to have fields that are not referenced, and no constraints are imposed on those fields.

Unless a schema definition is configured with `allow-unnamed-fields` set to `false`, it is only necessary to include information about fields whose values should be indexed or tokenized. However, it may be

desirable to define other fields that are expected in order to ensure that clients will not be permitted to store invalid values. See the *Working with JSON Indexes* for information about indexing JSON attributes.

As with standard LDAP schema, JSON schema constraints are enforced for any changes made after the constraints are defined. If there are already JSON values in the data before a JSON schema is defined for that attribute type (or before changes are made), values that already exist may violate those constraints. JSON schema constraints will also be enforced for data provided in an LDIF import, so that entries containing JSON objects that violate these constraints will be rejected.

JSON Field Constraints

Property	Description
`field`	Specifies the path to the target field as a string, with periods to separate levels of hierarchy. If any field name in the hierarchy itself includes a period, that period should be escaped with a backslash.
`value-type`	Specifies the expected data type for the target field. Values can include: ▪ `any`. The target field can have any value. ▪ `boolean`. The target field must have a value of true or false. ▪ `integer`. The target field must have a number that can be exactly represented as an integer. ▪ `null`. The target field must have a value of null. ▪ `number`. The target field must have a value that represents a valid JSON number ▪ `object`. The target field must have a value that represents a valid JSON object. The `allowed-fields` array may contain additional elements that define constraints for the fields that may be present in the object. ▪ `string`. The target field must have a value that represents a valid JSON string.
`is-required`	Specifies whether the target field is required to be present. If it is present, its value must be either `true` or `false`. If it is absent, a default of `false` is assumed.
`is-array`	Indicates whether the target field can be an array. If the value can be an array, all of the elements of the array must be of the type specified in the `value-type` field. It can be present with a value that is one of the single strings listed below. If it is absent, a value of `prohibited` is assumed. Values include: ▪ `required`. The target field must be an array with zero or more values of the specified type, and must not be a single value of the specified type. ▪ `optional`. The target field may be an array with zero or more values of the specified type, or it may be a single value of the specified type. ▪ `prohibited`. The target field must not be an array and may only be a single value of the specified type.
`allow-null-value`	Specifies whether the target field can have a value of `null` as an alternative to the specified `value-type`. If present, its value must be either `true` or `false`. If it is absent, a default of `false` is assumed. This has no effect if the value-type is `null`.
`allow-empty-object`	If the value of the target field is a JSON object (or an array of JSON objects), specifies whether empty objects are permitted. If present, the value must be either `true` (the object may have zero or more fields) or `false` (the object may have one or more fields). If it is absent, a default of `false` is assumed.

Property	Description
`index-values`	Specifies whether values of the target field should be indexed in backends. If present, the value must be either `true` (the field should be indexed) or `false` (the field should not be indexed). If it is absent, a default of `false` is assumed. If `true`, the `value-type` must be `boolean`, `integer`, `null`, or `string`.
`index-entry-limit`	Specifies the maximum number of entries in which a particular value may appear before the entry ID list for that value is no longer maintained. If present, the value must be an integer greater than or equal to one. If it is absent, the server will use the default index entry limit for the associated backend. This is only applicable if `index-values` is `true`.
`prime-index`	Specifies whether backends should prime the contents of the JSON index database into memory when they are opened. This is ignored if the backend's `prime-all-indexes` property has a value of `true`.
`cache-mode`	Specifies the cache mode to use for the contents of the JSON index database. If the value is not specified, the backend's default cache mode will be used. If a cache mode of `cache-keys-only` is configured, priming will only load the internal nodes into memory for the index. If `no-caching` is configured, no priming is performed for the index.
`tokenize-values`	Specifies whether values of the target field should be tokenized in backends. If present, the value is either `true` (field values should be tokenized) or `false`. If it is absent, a value of `false` is assumed. If `true`, the `value-type` must be `string`.
`allowed-values`	Specifies an explicit set of values that the target field is permitted to have. If present, the value must be an array of strings. Any attempt to use a value not in this array for the associated field is rejected. This can only be used with a `value-type` of `string`. If not present, any value can be used as long as it satisfies all other defined constraints.
`allowed-value-regular-expression`	Specifies a regular expression that the target field will be required to match. If present, the value must be a single string or an array of strings representing valid regular expressions (which may require escaping to represent in JSON). Any attempt to use a value that does not match one of the provided regular expressions will be rejected. This can only be used with a `value-type` of `string`. If not present, values are not required to match any regular expression.
`minimum-numeric-value`	Specifies the lower bound for values of the target field. If present, the value must be a single number (it can be an integer). Any attempt to use a value that is less than this number is rejected. This can only be used with a `value-type` of `integer` or `number`. If not present, no minimum numeric value applies.
`maximum-numeric-value`	Specifies the upper bound for values of the target field. If present, the value must be a single number (it can be an integer). Any attempt to use a value that is more than this number is rejected. This can only be used with a `value-type` of `integer` or `number`. If not present, no maximum numeric value applies.
`minimum-value-length`	Specifies the minimum number of characters allowed for values of the target field. If present, the value must be an integer. Any attempt to use a value with fewer characters than this number is rejected. This can only be used with a `value-type` of `string`. If not present, no minimum length applies.

Property	Description
`maximum-value-length`	Specifies the maximum number of characters allowed for values of the target field. If present, the value must be an integer. Any attempt to use a value with more characters than this number is rejected. This can only be used with a `value-type` of `string`. If not present, no maximum length applies.
`minimum-value-count`	Specifies the minimum number of elements that must be present in an array. If present, the value must be an integer. Any attempt to use an array with fewer elements is rejected. This can only be used with an `is-array` value of `required` or `optional`. If not present, no minimum array value count applies.
`maximum-value-count`	Specifies the maximum number of elements that must be present in an array. If present, the value must be an integer. Any attempt to use an array with more elements is rejected. This can only be used with an `is-array` value of `required` or `optional`. If not present, no maximum array value count applies.

ⓘ **Note:** When writing JSON objects in a local DB backend, field names and JSON primitive values of `null`, `true`, and `false` are always tokenized. Integers are either tokenized or compacted using their two's complement representation (other numbers are stored using string representations). Array and object sizes are compacted, and their contents are compacted based on their data types. String values are tokenized that match a recognizable format, including:

- Dates and times in common generalized time and ISO 8601 formats.
- UUIDs in which the alphabetic characters are either all uppercase or all lowercase.
- Strings of at least 12 bytes that are a valid base64 encoding.
- Strings of at least 6 bytes that are a valid hexadecimal encoding, in which the alphabetic characters are either all uppercase or all lowercase.

To Add Constraints to JSON Attributes

Steps

- Use the `dsconfig` tool (interactive or command line) to create and configure JSON attribute constraints. The following is a sample command:

```
$ bin/dsconfig create-json-attribute-constraints \
  --attribute-type appjson \
  --set enabled:true \
  --set allow-unnamed-fields:false
```

```
$ bin/dsconfig create-json-field-constraints \
  --attribute-type appjson \
  --json-field email.verified \
  --set value-type:boolean \
  --set is-required:true \
  --set index-values:true \
  --set tokenize-values:false \
  --set allow-null-value:true
```

```
$ bin/dsconfig create-json-field-constraints \
  --attribute-type appjson \
  --json-field email.type \
  --set value-type:string \
  --set is-required:true \
  --set index-values:true \
```

```
  --set tokenize-values:true \
  --set allowed-value:home \
  --set allowed-value:other \
  --set allowed-value:work \
  --set allow-null-value:false \
  --set minimum-value-length:1
```

```
$ bin/dsconfig create-json-field-constraints \
  --attribute-type appjson \
  --json-field email.value \
  --set value-type:string \
  --set is-required:true \
  --set index-values:true \
  --set tokenize-values:true \
  --set prime-index:true \
  --set allow-null-value:true \
  --set maximum-value-length:256 \
  --set minimum-value-length:1 \
  --set allowed-value-regular-expression:[-_\+\.\w\d]+@\w+\.\w{2,5}
```

Managing Password Policies

The PingDirectory Server provides a flexible password policy system to assign, manage, or remove password policies for root and non-root users. The password policy contains configurable properties for password expiration, failed login attempts, account lockout and other aspects of password and account maintenance on the Directory Server. The Directory Server also provides a global configuration option and a per-user privilege feature that disables parts of the password policy evaluation for production environments that do not require a password policy.

Viewing password policies

Password policies enforce a set of rules that ensure that access to data is not compromised through negligent password practices. The PingDirectory Server provides mechanisms to create and maintain password policies that determine whether passwords should expire, whether users are allowed to modify their own passwords, or whether too many failed authentication attempts should result in an account lockout. Many other options are available to fully configure a password policy for your PingData Platform system.

The Directory Server provides three out-of-the-box password policies that can be applied to your entries or as templates for configuring customized policies. The Default Password Policy is automatically applied to all users although it is possible to use an alternate password policy on a per-user basis. The Root Password Policy is enforced for the default root user, which uses a stronger password storage scheme (PBKDF2 rather than the salted 256-bit SHA-2 scheme) and also requires that a root user provide his or her current password to select a new password.

The Secure Password Policy provides a more secure option than the default policy that makes use of a number of features, including password expiration, account lockout, last login time and last login IP address tracking, password history, and a number of password validators.

> ⓘ **CAUTION:**
>
> Using the Secure Password policy as-is may notably increase write load in the server by requiring updates to password policy state attributes in user entries and/or requiring users to change passwords more frequently. In environments in which write throughput is a concern (including environments spread across multiple data centers requiring replication over a WAN), it may be useful to consider whether the policy should be updated to reduce the number of entry updates that may be required.

Viewing password policies

Steps

- You can view the list of password policies configured on the Directory Server using the **dsconfig** tool, in either interactive or non-interactive mode, or the Administrative Console. The following example demonstrates the process for obtaining a list of defined password policies in non-interactive mode:

```
$ bin/dsconfig list-password-policies
```

```
Password Policy      : Type : password-attribute : default-password-storage-
scheme
-----------------------:---------:--------------------:----------------------------
Default Password Policy : generic : userpassword    : Salted 256-bit SHA-2
Root Password Policy : generic : userpassword    : PBKDF2
Secure Password Policy  : generic : userpassword       : PBKDF2
```

Viewing a specific password policy

Steps

- Use **dsconfig** or the Administrative Console to view a specific password policy. In this example, view the Default Password Policy that applies to all uses for which no specific policy has been configured.

```
$ bin/dsconfig get-password-policy-prop \
  --policy-name "Default Password Policy"
```

```
Property                                 : Value(s)
-----------------------------------------:--------------------------
description                              : -
password-attribute                       : userpassword
default-password-storage-scheme          : Salted SHA-1
deprecated-password-storage-scheme       : -
password-validator                       : -
account-status-notification-handler      : -
allow-user-password-changes              : true
password-change-requires-current-password : false
force-change-on-add                      : false
force-change-on-reset                    : false
password-generator                       : Random Password Generator
require-secure-authentication            : false
require-secure-password-changes          : false
min-password-age                         : 0s
max-password-age                         : 0s
max-password-reset-age                   : 0s
password-expiration-warning-interval     : 5d
expire-passwords-without-warning         : false
allow-expired-password-changes           : false
grace-login-count                        : 0s
lockout-failure-count                    : 0s
lockout-duration                         : 0s
lockout-failure-expiration-interval      : 0s
require-change-by-time                   : -
last-login-time-attribute                : ds-pwp-last-login-time
last-login-time-format                   : -
previous-last-login-time-format          : -
idle-lockout-interval                    : 0s
password-history-count                   : 0s
password-history-duration                : 0s
```

About the password policy properties

The Directory Server provides a number of configurable properties that can be used to control password policy behavior.

ⓘ **Note:** To view a description of each of the password policy properties, see the *Ping Identity Directory Server Configuration Reference* that is bundled with the PingDirectory Server.

Some of the most notable properties include:

- **allow-user-password-changes**. Specifies whether users can change their own passwords. If a user attempts to change his/her own password, then the server will consult this property for the user's password policy, and will also ensure that the access control handler allows the user to modify the configured password attribute.
- **default-password-storage-scheme**. Specifies the names of the password storage schemes that are used to encode clear-text passwords for this password policy.
- **enable-debug**. When enabled, is used to debug password policy interaction. This property should be used in addition to the server's debug framework with a relevant debug target.
- **force-change-on-add**. Specifies whether users are required to change their passwords upon first authenticating to the Directory Server after their account has been created.
- **force-change-on-reset**. Specifies whether users are required to change their passwords after they have been reset by an administrator. An administrator is a user who has the `password-reset` privilege and the appropriate access control instruction to allow modification of other users' passwords.
- **idle-lockout-interval**. Specifies the maximum length of time that an account may remain idle (the associated user does not authenticate to the server) before that user is locked out. For accounts that do not have a last login time value, the password changed time or the account creation time will be used. If that information is not available, then the user will not be allowed to authenticate. It is strongly recommended that the server be allowed to run for a period of time with last login time tracking enabled (i.e., values for both `last-login-time-attribute` and `last-login-time-format` properties) to ensure that users have a last login time before enabling idle account lockout.
- **lockout-duration**. Specifies the length of time that an account is locked after too many authentication failures. The value of this attribute is an integer followed by a unit of seconds, minutes, hours, days, or weeks. A value of 0 seconds indicates that the account must remain locked until an administrator resets the password.
- **lockout-failure-count**. Specifies the maximum number of times that a user may be allowed to attempt to bind with the wrong password before that user's account becomes locked either temporarily (in which case the account will automatically be unlocked after a configurable length of time) or permanently (in which case an administrator must reset the user's password before the account may be used again). For example, if the value is set to `3`, the user is locked out after three failed attempts, even if a fourth attempt is made with the correct password.
- **max-password-age**. Specifies the maximum length of time that a user can continue to use the same password before he or she is required to choose a new password. The value can be expressed in seconds (s), minutes (m), hours (h), days (d), or weeks (w). A minimum length of time can also be specified before the user is allowed to change the password.
- **password-change-requires-current-password**. Specifies whether users must include their current password when changing their password. This applies for both password changes made with the password modify extended operation as well as simple modify operations targeting the password attribute. In the latter case, if the current password is required then the password modification must remove the current value and add the desired new value (providing both the current and new passwords in the clear rather than using encoded representations).
- **password-expiration-warning-interval**. Specifies the length of time before a user's password expires that he or she receives notification about the upcoming expiration (either through the password policy or password expiring response controls). The value can be expressed in seconds (s), minutes (m), hours (h), days (d), or weeks (w).

- **password-retirement-behavior**. Specifies the behavior of a password that is allowed a retirement period before becoming invalid. This setting may be used by application service accounts that require a transition period while updating passwords. This is disabled by default.
- **password-validator**. Specifies the names of the password validators that are used with the associated password storage scheme. The password validators are invoked when a user attempts to provide a new password, to determine whether the new password is acceptable.
- **require-secure-authentication**. Indicates whether users with the associated password policy are required to authenticate in a secure manner. This might mean either using a secure communication channel between the client and the server, or using a SASL mechanism that does not expose the credentials.
- **require-secure-password-changes**. Indicates whether users with the associated password policy are required to change their password in a secure manner that does not expose the credentials.

ⓘ **Note:** As an alternative to account lockout, a **`failed-bind-response-delay`** configuration property can be set on the LDAP Connection handler to instruct the server to introduce a delay (such as one second) into the process of returning a response to an unsuccessful bind operation. Delaying the response to a failed bind only affects the connection on which the bind was attempted, and still limits the rate at which a malicious client can try to guess a user's password. However, it will not affect other attempts to authenticate as that user on other connections, so the legitimate user can still authenticate with the correct password.

Access log

You can configure the server to maintain a recent login history for both successful and failed login attempts. This history can be maintained by count or duration, and you can configure the history separately for sucessful and failed login attempts.Each record in the login history contains the following:

- The time of the login attempt.
- The client IP address.
- The authentication method.
- The reason for failure for failed attempts.

You can optionally collapse information about multiple similar attempts on the same date to avoid flooding the history for accounts that bind frequently. Records have an additional attempt count that tracks the number of attempts with the same client IP address, authentications method, and failure reason on the same date. You can also configure the server to maintain each attempt separately, or to only update the history at most once per day.

Recent login history is disabled by default. You can enable and configure the recent login history in password policy. Recent login history can be retrieved with the **`get recent login history`** control, available in the LDAP SDK or with the **`ldapsearch`** and **`ldapmodfy`** commands. The recent login history is also available in the ds-pwp-state-json JSON attribute and in the password policy state extended operation and the **`manage-account`** command-line tool.

The recent login history can be enabled and configured with the following password policy configuration properties:

- maximum-recent-login-history-successful-authentication-count - The maximum number of records that the server will maintain about recent successful authentications.
- maximum-recent-login-history-successful-authentication-duration - The maximum length of time for which the server will maintain information about recent successful authentications.
- maximum-recent-login-history-failed-authentication-count - The maximum number of records that the server will maintain about recent failed authentication attempts.
- maximum-recent-login-history-failed-authentication-duration - The maximum length of time for which the server will maintain information about recent failed authentication attempts.
- recent-login-history-similar-attempt-behavior - The behavior that the server will exhibit for cases in which a user makes multiple authentication attempts on the same date in which all of the fields in the

record other than the timestamp (client-ip-address, authentication-method, and potentially failure-reason) match. Possible values for this property include the following:

- collapse-similar-attempts-on-the-same-date - Indicates that the server will only maintain one record for any given date with the same values for all non-timestamp fields, and it will use the additional-attempt-count field to keep track of the number of additional attempts that were collapsed into the same record. The timestamp field for that record will reflect the most recent attempt on that date. This will be the default behavior.
- maintain-every-attempt - Indicates that the server will maintain a separate record for every attempt, regardless of how similar it is to a previous attempt although duplicate attempts within the same millisecond may not be preserved (see *Replication considerations* on page 644).
- update-at-most-once-per-day - Indicates that the server will only maintain one record for any given date with the same values for all non-timestamp fields. This can help reduce the number of writes needed to maintain a recent login history, but the value of the timestamp field may not accurately reflect the timestamp of the most recent attempt.

None of these properties are defined by default. If at least one of these properties is defined, the server maintains a recent login history within the specified constraints.

If both the maximum-recent-login-history-successful-authentication-count and maximum-recent-login-history-successful-authentication-duration properties are defined, the server uses the more-restrictive value that applies to a given user. This is also true for the maximum-recent-login-history-failed-authentication-count and maximum-recent-login-history-failed-authentication-duration properties. For example, if the password policy is configured to maintain a successful count of 10 and a successful duration of 30 days, then a user who successfully authenticates on more than ten dates in a 30-day period would be capped at ten records, while a user who authenticates less frequently would only have records for however many attempts they made within those 30 days.

The server can collapse multiple authentication attempts from the same date into a single record if other fields in the record (client-ip-address, authentication-method, and potentially failure-reason, match, this caps the number of records that will be maintained if you want to maintain records by duration rather than count. However, because multiple records may be generated for the same user on the same date if something is different (for example., a different IP address or authentication method), there is technically no limit to the number of records that may be generated when using only a duration-based cap. To help mitigate this, even if you generally want to maintain a duration-based cap, you can specify a maximum count to place an upper bound on what informatoin the server will maintain for a given user.

ⓘ **Note:** The password policy state for a given user is only updated when that user attempts to authenticate to the server. Therefore, a user might have records in their entry for authentication attempts that occurred outside of the maximum duration if they have not made any authentication attempt within that duration. Further, if the server is configured to maintain recent login history for successful authentication attempts, then it will always keep a record of the most recent attempt (or an attempt from the same date as the most recent attempt if the recent-login-history-similar-attempt-behavior is set to update-at-most-once-per-day), even if that attempt occurred outside of the maximum duration. Similarly, if the server is configured to maintain a history of failed attempts, then it will always provide information about the most recent failed attempt, even if it is older than the maximum duration.

If the password policy is configured to maintain a recent login history, the ds-pwp-state-json virtual attribute includes a recent-login-history field whose value is a JSON object with the same representation used in the get recent login history response control. It may also include the following additional fields that provide information about related configuration in the password policy:

- maximum-recent-login-history-successful-authentication-count
- maximum-recent-login-history-successful-authentication-duration-seconds
- maximum-recent-login-history-failed-authentication-count
- maximum-recent-login-history-failed-authentication-duration-seconds

The password policy state extended operation provides support for two additional operations:

- An operation that may be used to retrieve the recent login history. The value returned in this operation will be a JSON object in the same format as used in the get recent login history response control and the ds-pwp-state-json virtual attribute.
- An operation that may be used to clear the recent login history for a user.

Replication considerations

Each login attempt is maintained as a separate attribute value to avoid the potential for data loss as a result of replication propagation delay. If a record of all login attempts were maintained within a single JSON object, the object written on one server would not reflect concurrent attempts made on other servers, and replication would only use what it perceives to be the most recent value rather than attempting to merge the values.

It is still possible that information about one or more attempts made around the same time could be lost. This includes the following cases:

- If the server is not configured to collapse information about multiple similar attempts, then it will not be able to record information about multiple similar attempts made within the same millisecond, because that would result in duplicate attribute values.
- If the server is configured to collapse information about multiple similar attempts, then concurrent modifications (especially on different servers) could cause the additional-attempt-count value to be incremented to the same value on each of those servers. If the concurrent attempts happened within the same millisecond, then only one of them would be preserved and any others would be lost. If the attempts did not occur within the same millisecond, then we may be able to infer the correct value during internal processing, but there may still be corner cases in which the server could lose information about one or more attempts.

Get Recent Login History control

A pair of request and response controls can be used to obtain the recent login history. The request control, which has an OID of 1.3.6.1.4.1.30221.2.5.61 and no value, may be included in a bind request to indicate that the server should return the recent login history in the bind response. This is provided in the response control, which has an OID of 1.3.6.1.4.1.30221.2.5.62 and a value containing only the string representation of a JSON object with the recent login history. That object will have either or both of two top-level fields:

- successful-attempts - This field ispresent if the server is configured to maintain a history of successful attempts, and its value will be an array of JSON objects with information about those successful attempts. In particular, each of those objects will contain the timestamp, client-ip-address, authentication-method, and additional-attempt-count fields as used in the ds-pwp-recent-login-history-json attribute.
- failed-attempts - This field is present if the server is configured to maintain a history of failed attempts. Its value is an array of JSON objects with information about these failed attempts. In particular, each of those objects contains the timestamp, client-ip-address, authentication-method, failure-reason, and additional-attempt-count fields as used in the ds-pwp-recent-login-history-json attribute.

The response control is only returned if the server is configured to maintain a recent login history. When provided, the elements of the arrays are arranged in chronological order from most-recent to least-recent.

The UnboundID LDAP SDK for Java provides support for these controls, including enhanced support for retrieving information from the response control value. However, by ensuring that the request control does not have a value and that the response control value is a simple string, this information should be readily accessible to applications using other APIs.

Modifying an existing password policy

About this task

You can modify an existing password policy to suit your company's requirements.

ⓘ **Note:** Password policies should be kept synchronized across all PingDirectory Server and Directory Proxy Server instances.

Steps

- Use **dsconfigdsconfig** in non-interactive mode: (in interactive or non-interactive mode) or the Administrative Console to modify the configuration for any defined password policy. The following example sets some of the properties presented in the previous section for the default password policy using

```
$ bin/dsconfig set-password-policy-prop \
  --policy-name "Default Password Policy" \
  --set "max-password-age:90 days" \
  --set "password-expiration-warning-interval:14 days" \
  --set "lockout-failure-count:3" \
  --set "password-history-count:6"
```

Creating new password policies

You can create any number of password policies in the Directory Server using either the **dsconfig** tool (in interactive or non-interactive mode) or the Administrative Console.

Creating a new password policy

Steps

- Use **dsconfig** (in interactive or non-interactive mode) or the Administrative Console to create a new pasword policy. The following example demonstrates the process for creating a new policy using **dsconfig** in non-interactive mode.

```
$ bin/dsconfig create-password-policy \
  --policy-name "Demo Password Policy" \
  --set "password-attribute:userpassword" \
  --set "default-password-storage-scheme:Salted SHA-256" \
  --set "force-change-on-add:true" \
  --set "force-change-on-reset:true" \
  --set "password-expiration-warning-interval:2 weeks" \
  --set "max-password-age:90 days" \
  --set "lockout-duration:24 hours" \
  --set "lockout-failure-count:3" \
  --set "password-change-requires-current-password:true"
```

Assigning a password policy to an individual account

About this task

To indicate that a user should be subject to a particular password policy (rather than automatically inheriting the default policy), include the `ds-pwp-password-policy-dn` operational attribute in that user's entry with a value equal to the DN of the desired password policy for that user. This attribute can be explicitly included in a user's entry, or it can be generated by a virtual attribute, which makes it easy to apply a custom password policy to a set of users based on a flexible set of criteria.

Steps

1. Create a file (`assign.ldif`) with the following contents:

```
dn: uid=user.1,ou=People,dc=example,dc=com
changetype: modify
add: ds-pwp-password-policy-dn
```

```
ds-pwp-password-policy-dn: cn=Demo Password Policy,cn=Password
 Policies,cn=config
```

2. Use **ldapmodify** to apply the modification to the user's entry.

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

Assigning a password policy using a virtual attribute

About this task

It is possible to automatically assign a custom password policy for a set of users using a virtual attribute. The virtual attribute can be configured so that it can use a range of criteria for selecting the entries for which the virtual attribute should appear.

Steps

1. Create an LDIF file, which may be used to add a group to the server.

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

dn: cn=Engineering Managers,ou=groups,dc=example,dc=com
objectClass: groupOfUniqueNames
objectClass: top
cn: Engineering Managers
uniqueMember: uid=user.0,ou=People,dc=example,dc=com ou: groups
```

2. Use **ldapmodify** to add the entries to the server.

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

3. Use **dsconfig** to create a virtual attribute that will add the `ds-pwp-pasword-policy-dn` attribute with a value of `cn=Demo Password Policy,cn=Password Policies,cn=config` to the entries for all users that are members of the `cn=Engineering Managers,ou=Groups,dc=example,dc=com` group.

```
$ bin/dsconfig create-virtual-attribute \
  --name "Eng Mgrs Password Policy" \
  --type user-defined \
  --set "description:Eng Mgrs Grp PWPolicy" \
  --set enabled:true \
  --set attribute-type:ds-pwp-password-policy-dn \
  --set "value:cn=Demo Password Policy,cn=Password Policies,cn=config" \
  --set "group-dn:cn=Engineering Managers,ou=Groups,dc=example,dc=com
```

4. Use **ldapsearch** to verify that a user in the group contains the assigned password policy DN.

```
$ bin/ldapsearch --baseDN dc=example,dc=com "(uid=user.0)" \
ds-pwp-password-policy-dn
```

```
dn: uid=user.0,ou=People,dc=example,dc=com
ds-pwp-password-policy-dn: cn=Demo Password Policy,cn=Password
 Policies,cn=config
```

Deleting a password policy

About this task

You can delete a password policy using either the **dsconfig** tool (in interactive or non-interactive mode) or the Administrative Console.

Steps

- Run the **dsconfig** command with the `delete-password-policy` subcommand to remove a password policy.

```
$ bin/dsconfig delete-password-policy \
  --policy-name "Demo Password Policy"
```

Modifying a user's password

There are two primary ways to change user passwords in the Directory Server:

- Perform a modify operation which replaces the value of the password attribute (often `userPassword`). Note that in some configurations, when a user attempts to change his or her own password it may be necessary to perform the modification by removing the password value and adding the desired new value to demonstrate that the user knows the current password value.
- Use the password modify extended operation to change the password. Note that if a user is changing his or her own password, it may be necessary to provide the current password value. The server will allow a new password to be provided (assuming that the new password is acceptable to all configured password validators), or it may automatically generate a new password for the user.

Note that regardless of the mechanism used to change the password, all password values should be provided in clear text rather than pre-encoded, and the user will be required to have sufficient access control rights to update the password attribute in the target user's entry. Further, when one user attempts to change the password for another user, then the requester will be required to have the `password-reset` privilege.

Validating a password

The requirements that the server will impose for a password change can be displayed to users. The get password quality requirements extended operation can be used to retrieve information about the requirements, which can then be sent to an end user before an attempted password change. These requirements can also be used to enable client-side validation, so that any password problems can be identified before it is sent to the server. The password validation details request control can be included in an add or modify request, or a password modify extended request, to identify which validation requirements were not met by the password provided in the request.

Password validators can be configured with user-friendly messages that describe the password requirements, and the messages that should be returned if a proposed password does not satisfy those requirements. The server will generate these messages if they are not provided in the configuration.

Password validator properties include **validator-requirement-description** and **validator-failure** message. The following is a simple password validator configuration that requires passwords to contain a minimum of five characters, and lists custom validator messages:

```
$ dsconfig create-password-validator \
  --validator-name "Minimum 5 Characters Password Validator" \
  --type length-based --set enabled:true \
  --set "validator-requirement-description:The password must contain
         at least 5 characters." \
  --set "validator-failure-message:The password did not contain
         at least 5 characters." \
  --set min-password-length:5
```

After the password validator is created, it should be assigned to a Password Policy to take affect:

```
$ dsconfig set-password-policy-prop \
  --policy-name "Default Password Policy" \
  --set "password-validator:Minimum 5 Characters Password Validator"
```

When a user authenticates, password validation processing is performed so that the server has access to the user's clear-text password. Password properties include the following:

- bind-password-validator - Specifies which validators to invoke on bind.
- password-validator - Specifies which validators to invoke during a password change.
- minimum-bind-password-validation-frequency — Specifies how frequently the server should validate a user's password during bind. Although you can specify that the password should be validated during each bind, it is probably sufficient to only do so periodically (for example, once a week or once a month).
- bind-password-validation-failure-action — Specifies the action the server should take if a user's password fails validation. By default, the account will be placed in a "must change password" state where the user is allowed to bind but any other operations the user attempts will fail until the user changes their password. Alternatively, the account can be locked so that the password needs to be reset by an administrator, or the server can generate an account status notification to recommend that the user choose a new password.

Retiring a password

An account password can be retired and rotated out of service, instead of immediately invalidated. This enables a new password to be assigned to an account while keeping the original password valid for a period of time to enable a transition. This is useful for application service accounts that require uninterrupted authentication with the server.

This behavior is disabled by default, but can be enabled in the password policy configuration by setting the **password-retirement-behavior** and **maximum-retired-password-age** properties.

To manually retire an account password or purge a password that has been retired, use the **ldapmodify** and **ldappasswordmodify** commands with options **-- retireCurrentPassword** and **--purgeCurrentPassword**. To use these commands on an account, the password policy that governs the account must have the **password-retirement-behavior** enabled.

Changing a user’s password using the Modify operation

Steps

- Use **ldapmodify** to change a user’s password by replacing the `userPassword` attribute.

```
$ bin/ldapmodify
dn: uid=user.0,ou=People,dc=example,dc=com
changetype: modify
replace: userPassword
userPassword: newpw
```

Changing a user's password using the Password Modify extended operation

Steps

- Use **ldappasswordmodify** to request that the Password Modify extended operation be used to modify a user’s password.

```
$ bin/ldappasswordmodify --authzID dn:uid=jdoe,ou=People,dc=example,dc=com
 \
  --newPassword newpw
```

Using an automatically-generated password

Steps

- Use **`ldappasswordmodify`** to automatically generate a new password for a user.

```
$ bin/ldappasswordmodify --authzID "u:user.1"

The LDAP password modify operation was successful
Generated Password: fbi27oqy
```

Enabling YubiKey authentication

Users can be enabled to authenticate with YubiKey devices (available from Yubico), which generate secure one-time passwords, with the UNBOUNDID-YUBIKEY-OTP SASL mechanism. A YubiKey device generates a different password for every authentication attempt, and that one-time password is sent to a validation service to ensure that it is genuine and has not been used in an earlier authentication attempt. Although it is possible to use this one-time password as the only proof of identity, it is typically combined with a static password as a form of two-factor authentication.

YubiKey authentication requires server configuration and the addition of this capability to a user entry. Configuration of a client ID and API key to use when communicating with the validation service is also required. The API key is a shared secret between the YubiKey validation service and the client that is interacting with it, and is used when generating digital signatures so that both the server and the YubiKey validation service can ensure that the peer server is genuine.

All server and user entry configuration details are available in the *Security Guide*.

Enabling social login

Authentication involving credentials that do not reside in, or cannot be forwarded to or validated by the Directory Server (such as social login through Facebook, Google, or Twitter) can be enabled with the the UNBOUNDID-EXTERNALLY-PROCESSED-AUTHENTICATION SASL mechanism. The bind request will not include any credentials, and authentication with this mechanism will not actually change the state of the underlying client connection. The server will behave as if the bind request included the retain identity request control, regardless of whether or not that control was included.

Bind requests using this mechanism can include any request controls that are permitted with other bind requests. If the externally-processed authentication is successful, the client can include the get password policy state issues request control in the bind request to obtain information about any password policy state issues that can cause the Directory Server authentication attempt to fail. The password policy request control can also be included to obtain certain password policy state warnings and errors, or to look for the password expired/password expiring controls in the bind response.

All server and user entry configuration details are available in the *PingDirectory Server Security Guide*.

Managing user accounts

The Directory Server provides a user management utility, the **`manage-account`** tool, that provides a means to quickly view and manipulate a number of password and account policy properties for a user or group of users.

> ⓘ **Note:** A disabled account status (for example, `account-is-disabled: true`) is different from an *account lockout* due to password policy. Unlocking a user account can be done with the **`manage-account`** tool. A disabled account requires the administrator to enable the account; password resets are not involved.

PingDirectory Server also hosts the Self Service Account Manager project at https://github.com/pingidentity/ssam, which is a customizable web application allowing users to perform their own account

registration, profile updates, and password changes. The project is for testing and development purposes, and is not a supported PingDirectory Server application.

Returning the password policy state information

Steps

- Use **manage-account** to get information about the account's password policy.

```
$ bin/manage-account get-all \
  --targetDN uid=user.1,ou=People,dc=example,dc=com
```

```
Password Policy DN: cn=Demo Password Policy,cn=Password Policies,cn=config
Account Is Disabled: false
Account Expiration Time:
Seconds Until Account Expiration:
Password Changed Time: 19700101000000.000Z
Password Expiration Warned Time:
Seconds Until Password Expiration: 1209600
Seconds Until Password Expiration Warning: 0
Authentication Failure Times:
Seconds Until Authentication Failure Unlock:
Remaining Authentication Failure Count: 3
Last Login Time:
Seconds Until Idle Account Lockout:
Password Is Reset: false
Seconds Until Password Reset Lockout:
Grace Login Use Times:
Remaining Grace Login Count: 0
Password Changed by Required Time:
Seconds Until Required Change Time:
Password History:
```

Determining whether an account is disabled

Steps

- Use **manage-account** to determine whether a user's account has been disabled.

```
$ bin/manage-account get-account-is-disabled \
  --targetDN uid=user.1,ou=People,dc=example,dc=com
```

```
Account Is Disabled: true
```

Disabling an account

Steps

- Use **manage-account** to disable a user's account.

```
$ bin/manage-account set-account-is-disabled \
  --operationValue true --targetDN uid=user.1,ou=People,dc=example,dc=com
```

```
Account Is Disabled: true
```

Enabling a disabled account

Steps

- Use **`manage-account`** to enable a user's account.

```
$ bin/manage-account clear-account-is-disabled \
  --targetDN uid=user.1,ou=People,dc=example,dc=com
```

```
Account Is Disabled: false
```

Assigning the manage-account access privileges to non-root users

About this task

Non-root users (e.g., `uid=admin`) with admin right privileges require access control permission to interact with certain password policy operational attributes when using the **`manage-account`** tool.

For example, the presence of the `ds-pwp-account-disabled` operational attribute in an entry determines that the entry is disabled. If the non-root admin user does not have the access privilege to read or interact with the `ds-pwp-account-disabled` operational attribute, the **`manage-account`** tool may report that the account is active. An account is considered active if the `ds-pwp-account-disabled` operational attribute does not exist in the entry or if the admin user does not have permission to see it.

Use the following procedure to give access rights to the non-root admin user.

Steps

1. Create a non-root user admin account, such as `uid=admin,dc=example,dc=com`. Grant the `password-reset` privilege to the account. See step 1 and 6 in the Configuring Administrators section for more information.
2. Run the **`manage-account`** tool to view the account status for an account.

```
$ bin/manage-account get-all \
  --targetDN uid=user.0,ou=People,dc=example,dc=com
```

```
Password Policy DN:  cn=Default Password Policy,cn=Password
 Policies,cn=config
Account Is Disabled:  false
Account Expiration Time:
Seconds Until Account Expiration:
Password Changed Time:  19700101000000.000Z
Password Expiration Warned Time:
Seconds Until Password Expiration:
Seconds Until Password Expiration Warning:
Authentication Failure Times:
Seconds Until Authentication Failure Unlock:
Remaining Authentication Failure Count:
Last Login Time:
Seconds Until Idle Account Lockout:
Password Is Reset:  false
Seconds Until Password Reset Lockout:
Grace Login Use Times:
Remaining Grace Login Count:  0
Password Changed by Required Time:
Seconds Until Required Change Time:
Password History:
```

3. Grant access control privileges to an account. The following allows access to manage accounts to a helpdesk user. Depending on the configuration requirements, this user may also need the `permit-get-password-policy-state-issues` and `password-reset` privileges.

```
dn: dc=example,dc=com
 changetype: modify
 add: aci
 aci: (targetattr="userPassword||ds-pwp-last-login-time||ds-pwp-password-
changed-by-required-time||ds-pwp-reset-time||ds-pwp-warned-time||
 ds-pwp-account-disabled||ds-pwp-account-expiration-time||ds-pwp-password-
policy-dn||ds-pwp-auth-failure||ds-pwp-last-login-ip-address||
 ds-pwp-retired-password||ds-pwp-account-activation-time||pwdReset||
pwdChangedTime||pwdAccountLockedTime")
 (version 3.0; acl "Grant full access to PWP related attributes to
 helpdesk"; allow (all) userdn="ldap:///uid=helpdesk,dc=example,dc=com";)
```

4. Run the **`manage-account`** tool to disable an account. The following command sets the `account-is-disabled` property to true for the `uid=user.0,dc=example,dc=com`.

```
$ bin/manage-account set-account-is-disabled \
  --targetDN uid=user.0,ou=People,dc=example,dc=com \
  --operationValue true
```

```
Account Is Disabled:  true
```

5. Run the **`ldapsearch`** tool to view the presence of the `ds-pwp-account-disabled` operational attribute in the entry.

```
$ bin/ldapsearch --baseDN dc=example,dc=com "(uid=user.0)" "+"
```

```
dn: uid=user.0,ou=People,dc=example,dc=com
ds-pwp-account-disabled: true
```

Disabling password policy evaluation

The Directory Server provides a global configuration property (`disable-password-policy-evaluation`) that can be used to disable most password policy evaluation processing. This provides a convenience for those production environments that do not require password policy support. If the `disable-password-policy` property is set to true, passwords will still be encoded and evaluated, but only account expiration and account disabling will be in effect. All other password policy properties, such as password expiration, lockout, and force change on add or reset, are ignored.

The server also supports the use of a `bypass-pw-policy` privilege, which can be used to skip password policy evaluation for operations on a per-user basis. If a user has this privilege, then they will be allowed to perform operations on user entries that would normally be rejected by the password policy associated with the target entry. Note that this privilege will not have any effect for bind operations.

Globally disabling password policy evaluation

Steps

- Use **`dsconfig`** to set the `disable-password-policy-evaluation` property globally for the Directory Server.

```
$ bin/dsconfig --no-prompt set-global-configuration-prop \
  --set "disable-password-policy-evaluation:false"
```

Exempting a user from password policy evaluation

Steps

- Use **`ldapmodify`** to add the `bypass-pw-policy` privilege to a user entry.

```
$ bin/ldapmodify
dn: uid=user.1,ou=People,dc=example,dc=com
changetype: modify
add: ds-privilege-name
ds-privilege-name: bypass-pw-policy
```

About managing password validators

A password validator is a password policy component that is used to determine if a new password is acceptable. A password policy can be configured with any number of password validators. If a password policy is configured with multiple password validators, then all of them must consider a proposed new password acceptable before it will be allowed.

Password validators

The PingDirectory Server offers a number of types of password validators, including those listed in the following table. It is also possible to use the Server SDK to create custom password validators with whatever constraints are necessary for your environment.

Password Validators

Password Validators	Description
Attribute Value	Ensures that the proposed password does not match the value of another attribute in the user's entry. The validator can be configured to look in all attributes or in a subset of attributes. It can perform forward and reverse mapping, and it can also reject values which are substrings of another attribute.
Character Set	Ensures that the proposed password contains a sufficient number of characters from one or more user-defined character sets. For example, the validator can ensure that passwords must have at least one lowercase letter, one uppercase letter, one digit, and one symbol.
Commonly-Used Passwords Dictionary	Ensures that the proposed password is not one of 10,000 commonly used passwords. These are words that are common for attackers to use when trying to access user accounts. The Commonly-Used Passwords validator is invoked by the Secure Password Policy by default. The word list is located in `<server-root>/config/commonly-used-passwords.txt`, and can be used to create a custom validator, but should not be modified.
Dictionary	Ensures that the proposed password is not present in a specified dictionary file, optionally also testing the password with all characters in reverse order. A large dictionary file is provided with the server, but the administrator can supply an alternate dictionary. In this case, then the dictionary must be a plain-text file with one word per line.
Haystack Password Validator	Ensures that the proposed password is secure based on a combination of its length and the types of characters that it contains. For example, a longer password containing only lowercase letters may be stronger than a shorter password containing a mix of uppercase and lowercase letters, numbers, and symbols. This is based on the Gibson Research Corporation Password Haystacks concept.

Password Validators	Description
Length-Based Password Validator	Ensures that the number of characters in the proposed new password is within an acceptable range. Both a maximum and minimum number of characters may be specified.
Regular Expression Validator	Ensures that a proposed password either matches or does not match a given regular expression.
Repeated Characters	Ensures that a proposed password does not contain a substring in which the same character is repeated more than a specified number of times (for example, "aaaaa" or "aaabbb"). The validator can be configured to operated in a case-sensitive or case-insensitive manner, and you can also define custom sets of equivalent characters (for example, you could define all digits as equivalent, so the proposed password could not contain more than a specified number of consecutive digits.
Similarity-Based Password Validator	Ensures that the proposed new password is not too similar to the current password, using the Levenstein Distance algorithm, which calculates the number of characters that need to be inserted, removed, or replaced to transform one string into another. Note that for this password validator to be effective, it is necessary to have access to the user's current password. Therefore, if this password validator is to be enabled, the `password-change-requires-current-password` attribute in the password policy configuration must also be set to true.
Unique Characters	Ensures that the proposed password contains at least a specified minimum number of unique characters, optionally using case-insensitive validation.

Configuring password validators

You can use the **`dsconfig`** configuration tool or the Administrative Console to configure or modify any password validators. Once you have defined your password validators, you can add them to an existing password policy. The following example procedures show the dsconfig non-interactive commands necessary to carry out such tasks. If you use **`dsconfig`** in interactive command-line mode, you can access the Password Validator menu in the Basic Objects menu. For more details on the password validator properties, see the *PingDirectory Server Configuration Reference*.

Viewing the list of defined password validators

Steps

- Use **`dsconfig`** to view the set of password validators defined in the Directory Server.

Configuring the Attribute Value Password Validator

Steps

1. Use **`dsconfig`** to edit the existing default configuration for the Attribute Value Password Validator. For example, the following change configures the validator to only examine a specified set of attributes..

```
$ bin/dsconfig set-password-validator-prop \
  --validator-name "Attribute Value" \
  --set match-attribute:cn \
  --set match-attribute:sn \
  --set match-attribute:telephonenumber \
  --set match-attribute:uid
```

2. Update an existing password policy to use the Attribute Value Password Validator.

```
$ bin/dsconfig set-password-policy-prop \
  --policy-name "Default Password Policy" \
  --set "password-validator:Attribute Value"
```

3. Test the Attribute Value Password Validator by submitting a password that is identical to one of the configured attributes (`cn`, `sn`, `telephonenumber`, `uid`).

```
$ bin/ldappasswordmodify --authzID "uid=user.0,ou=People,dc=example,dc=com"
 \
  --newPassword user.0
```

```
The LDAP password modify operation failed with result code 53
Error Message: The provided new password failed the validation checks
 defined in the
server: The provided password was found in another attribute in the user
 entry
```

Configuring the Character Set Password Validator

Steps

1. Use **`dsconfig`** to edit the existing default configuration. In this example, we change the requirement for special characters making them optional in a password, and add a requirement that at least two digits must be included in the password. Thus, in this example, all newly created passwords must have at least one lowercase letter, one uppercase letter, two digits, and optionally any special characters listed.

```
$ bin/dsconfig set-password-validator-prop \
  --validator-name "Character Set" \
  --remove character-set:1:0123456789 \
  --remove "character-set:1:~\!@#\$\%^&*()-_=+[]{}\|;:,.<>/?" \
  --add character-set:2:0123456789 \
  --add "character-set:0:~\!@#\$\%^&*()-_=+[]{}\|;:,.<>/?" \
  --set allow-unclassified-characters:false
```

2. Update an existing password policy to use the Character Set Password Validator.

```
$ bin/dsconfig set-password-policy-prop \
  --policy-name "Default Password Policy" \
  --set "password-validator:Character Set"
```

3. Test the Character Set Password Validator by submitting a password that meets the requirements (one lowercase letter, one uppercase letter, two digits). The following example should reject the given password because it does not pass the Character Set Password Validator.

```
$ bin/ldappasswordmodify \
  --authzID "uid=user.0,ou=People,dc=example,dc=com" --newPassword abab1
```

Configuring the Length-Based Password Validator

Steps

1. Use **`dsconfig`** to edit the existing default configuration. In this example, we set the required minimum number of characters in a password to five.

```
$ bin/dsconfig create-password-validator \
  --validator-name "Length-Based Password Validator" \
  --set max-password-length:5 --set min-password-length:5
```

2. Update an existing password policy to use the Length-Based Password Validator.

```
$ bin/dsconfig set-password-policy-prop \
  --policy-name "Default Password Policy" \
```

```
  --set "password-validator:Length-Based Password Policy"
```

3. Test the Length-Based Password Validator by submitting a password that has fewer than the minimum number of required characters.

```
$ bin/ldappasswordmodify \
  --authzID "uid=user.0,ou=People,dc=example,dc=com" --newPassword abcd
```

```
The LDAP password modify operation failed with result code 53
Error Message: The provided new password failed the validation checks
 defined in
the server: The provided password is shorter than the minimum required
 length of
5 characters
```

Configuring the Regular Expression Password Validator

Steps

1. Use **`dsconfig`** to create a Regular Expression password validator. The following password validator checks that the password contains at least one number, one lowercase letter, and one uppercase letter with no restrictions on password length. If the password matches the regular expression, then it will be accepted. When using the following command, remember to include the LDAP/LDAPS connection parameters (host name and port), bind DN, and bind password.

```
$ bin/dsconfig create-password-validator \
  --validator-name "Regular Expression" \
  --type regular-expression --set enabled:true \
  --set "match-pattern:^\\w*(?=\\w*\\d)(?=\\w*[a-z])(?=\\w*[A-Z])\\w*\$" \
  --set match-behavior:require-match
```

2. Update an existing password policy to use the Regular Expression validator.

```
$ bin/dsconfig set-password-policy-prop \
  --policy-name "Default Password Policy" \
  --set "password-validator:Regular Expression"
```

3. Test the Regular Expression Validator by submitting a password that meets the requirements (contains one number, one lowercase letter, and one uppercase letter), then run it again with a password that does not meet these requirements.

```
$ bin/ldappasswordmodify \
  --authzID "uid=user.0,ou=People,dc=example,dc=com" --newPassword baaA1
```

```
The LDAP password modify operation was successful
```

4. Try another password. The following password should fail, because no uppercase letter is present.

```
$ bin/ldappasswordmodify \
  --authzID "uid=user.0,ou=People,dc=example,dc=com" --newPassword baaa1
```

```
Error Message: The provided new password failed the validation checks
defined in the server: The provided password is not acceptable because it
 does
not match regular expression pattern '^\w*(?=\w*\d)(?=\w*[a-z])(?=\w*[A-
Z])\w*$'
```

Configuring the Repeated Character Password Validator

Steps

1. Use **dsconfig** to edit the existing default configuration.
   - In this example, we set the maximum consecutive length of any character to 3. For example, the following validator rejects any passwords, such as "baaaa1" or "4eeeeb", etc.

     ```
     $ bin/dsconfig set-password-validator-prop \
       --validator-name "Repeated Characters" \
       --set max-consecutive-length:3
     ```
   - Or, you can configure the validator to reject any character from a pre-defined character set that appears more than the specified number of times in a row (2). You can also specify more than one character set. For example, the following validator defines two characters sets: [abc] and [123]. It rejects any passwords with more than two consecutive characters from a character set. Thus, "aaa", "bbb", "ccc", "abc", or "123" and so on fails, but "12a3" is okay.

     ```
     $ bin/dsconfig set-password-validator-prop \
       --validator-name "Repeated Characters" \
       --set character-set:123 --set character-set:abc
     ```
2. Update an existing password policy to use the Repeated Character Password Validator.

   ```
   $ bin/dsconfig --no-prompt set-password-policy-prop \
     --policy-name "Default Password Policy" \
     --set "password-validator:Repeated Characters"
   ```
3. Test the Repeated Character Validator by submitting a password that has more than the maximum allowable length of consecutive characters.

   ```
   $ bin/ldappasswordmodify \
     --authzID "uid=user.0,ou=People,dc=example,dc=com" \
     --newPassword baaa1
   ```

   ```
   The LDAP password modify operation failed with result code 53
   Error Message: The provided new password failed the validation checks
    defined
   in the server: The provided password contained too many instances of the
    same
   character appearing consecutively. The maximum number of times the same
   character may appear consecutively in a password is 2
   ```

Configuring the Similarity-Based Password Validator

Steps

1. Use **dsconfig** to edit the existing default configuration. In this example, we set the minimum number of differences to 2.

   ```
   $ bin/dsconfig set-password-validator-prop \
     --validator-name "Similarity-Based Password Validator" \
     --set min-password-difference:2
   ```
2. Update an existing password policy to use the Similarity-Based Password Validator. The `password-change-requires-current-password` property must be set to TRUE, so that the password policy will ensure that the user's current password is available when that user is choosing a new password.

   ```
   $ bin/dsconfig set-password-policy-prop \
     --policy-name "Default Password Policy" \
     --set "password-validator:Similarity-Based Password Validator" \
     --set password-change-requires-current-password:true
   ```

3. Test the Similarity-Based Password Validator by submitting a password that has fewer than the minimum number of changes (e.g., 2). The **`ldappasswordmodify`** command requires the `--currentPassword` option when testing the Similarity-Based Password Validator.

```
$ bin/ldappasswordmodify \
  --authzID "uid=user.0,ou=People,dc=example,dc=com" \
  --currentPassword abcde --newPassword abcdd

The LDAP password modify operation failed with result code 49
```

Configuring the Unique Characters Password Validator

Steps

1. Use **`dsconfig`** to edit the existing default configuration. In this example, we set the minimum number of unique characters that a password is allowed to contain to 3.

```
$ bin/dsconfig set-password-validator-prop \
  --validator-name "Similarity-Based" --set min-unique-characters:3
```

2. Update an existing password policy to use the Unique Characters Password Validator.

```
$ bin/dsconfig set-password-policy-prop \
  --policy-name "Default Password Policy" \
  --set "password-validator:Unique Characters"
```

3. Test the Unique Characters Password Validator by submitting a password that has fewer than the minimum number of unique characters (e.g., 3).

```
$ bin/ldappasswordmodify \
  --authzID "uid=user.0,ou=People,dc=example,dc=com" \
  --newPassword aaaaa

The LDAP password modify operation failed with result code 53
Error Message: The provided new password failed the validation checks
 defined
in the server: The provided password does not contain enough unique
 characters.
The minimum number of unique characters that may appear in a user password
 is 3
```

Managing Replication

The PingDirectory Server provides a robust replication system to ensure high availability and fast failover in production environments. Write requests can be handled by every server in the topology with the replication component performing immediate synchronization with other members. The replicated server environment ensures that LDAP clients can seamlessly fail over to another server instance.

This chapter presents the architectural overview of replication, detailed configuration steps, ways to monitor replication as well as troubleshooting steps.

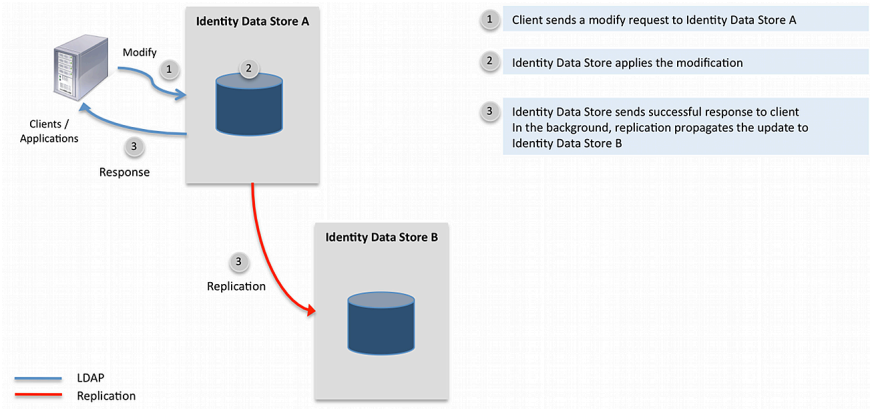
Overview of replication

Replication is a data synchronization mechanism that ensures that updates made to a database are automatically replayed to other servers. Replication improves data availability when unforeseen or planned outages occur, and improves search performance by allowing client requests to be distributed across multiple servers.

By default, all Directory Servers participating in replication are writable, so that LDAP clients can perform updates at any of these Directory Server instances. These updates will be propagated to the other servers automatically in the background and applied in the same order as the updates were entered. The

replication process flow is designed to immediately propagate changes to the other replicas in the topology with little or no latency.

The following figure demonstrates the basic flow of replication.



MY TITLE Replication Process Flow

The benefits of replication can be summarized as follows:

- **High-Availability**. Because the data is fully replicated on all other servers in the topology, replication allows participating servers to process all types of client requests. This mitigates any availability issues when a particular server is down due to a planned maintenance or unplanned outage. For those servers that are temporarily unavailable, they will receive updates when they become available again.
- **Improved Search Performance**. Search requests may be directed to any Directory Server participating in replication, which improves search performance over systems that only access single servers. Note, however, that replication does not improve write throughput since updates need to be applied at all servers.
- **WAN Friendly Data Synchronization**. The built-in compression feature in the replication protocol allows efficient propagation of updates over WAN links.

Replication versus synchronization

Replication is not a general purpose synchronization facility as it creates replicas with exact copies of the replicated data. Synchronization, on the other hand, can transform data between two different Directory Information Tree (DIT) structures, map attribute types, synchronize subsets of branches and specific object classes. The differences between replication and synchronization are illustrated as follows:

- **Replication cannot Synchronize between Different DIT Structures**. The DN of replicated entries must be the same on all servers. In some situations, it may be desirable to replicate entries with the help of DN mapping that are under different base DNs, but represent the same data, for example `uid=john.doe,ou=people,o=corp` on one server may represent the same user as `uid=john.doe,ou=people,dc=example,dc=com`. This is not supported by replication. Synchronization fully supports this feature.
- **Replication cannot Map Attribute Types or Transform Attribute Values**. In some situations, it may be necessary to map attribute types or transform attribute values when synchronizing data from one server to another. Replication does not support either attribute type mappings or attribute value transformations.
- **Replication does Not Support Fractional Replication**. Replication cannot be configured to replicate a subset of the attribute types from the replicated data set. Synchronization fully supports this feature.
- **Replication does Not Support Sparse Replication**. Replication cannot be configured to replicate entries with a particular object class only. Synchronization fully supports this feature.
- **Replication Requires Full Control of Replicated Data**. When two servers participate in replication, both servers implicitly trust each other using public key cryptography and apply all updates received via replication, which is considered an internal operation. While trust between servers is established

between two endpoint servers, synchronization does not require full control of the data. Disparate server system endpoints can be synchronized, such as a PingDirectory Server and a RDBMS database endpoint with each fully in control of its own data.

If replication does not meet your data synchronization requirements, consider using PingDataSync Server, which provides the versatility and robust performance required for most production environments.

Replication terminology

The following replication terms are used throughout this chapter.

Replication Terminology

Term	Description
Assured Replication	For applications that require immediate access to replicated data or require data consistency over response time, administrators can configure *assured replication*, where data is guaranteed to replicate *before* the response is returned to the client.
Change Number	A 64-bit number used to consistently order replication updates across the entire topology. It is composed of 3 fields: the timestamp of the update (measured in milliseconds since 00:00:00 UTC, January 1, 1970), the Replica ID, and the Sequence Number.
Conflict	Client updates made at different replicas affecting the same entry may be found in conflict when the updates are replayed at another replica. The Change Number of each update allows most of these conflicts to be resolved automatically. Certain updates, such as adding an entry with different attribute values at two servers simultaneously, result in conflicts that are flagged for required manual resolution.
Eventual Consistency	When not using assured replication, recent updates from LDAP clients are not immediately present at all servers. Out-of-sync data will be eventually synchronized and will be consistent on all servers. The network latency typically controls how long a given update takes to replicate.
Global Administrator	The administrative user with full rights to manage the replication topology. The user is created at the time of replication enable between two non-replicating servers and thereafter copied to newly enabled servers. The replication command-line utility expects the user name of the Global Administrator (by default, admin). The user is stored in `cn=Topology Admin Users,cn=Topology,cn=config`.
Historical Data	Historical Data are records of attribute value changes as a result of an LDAP Modify Operation. Historical Data is stored in the `ds-sync-hist` attribute of the target entry. This information allows replication to resolve conflicts from simultaneous LDAP Modify operations automatically.
Location	The collection of servers that may have similar performance characteristics when accessed from this Directory Server or reside in the same data center or city. A separate location setting may be defined for each data center, such as Austin, London, Chicago, etc. Location settings are used in the selection of the WAN Gateway server.
Modify Conflict	A conflict between two LDAP Modify operations. Conflicts arising from LDAP Modify operations are automatically resolved.

Term	Description
Naming Conflict	Any conflict other than Modify Conflicts. Naming conflicts typically include an operation that changes the DN of the target entry, creates an entry, or deletes an entry at one Replica.
Replica	The component in the Directory Server that handles interaction with a single replication domain, for example, the `dc=example,dc=com` replication domain within the `userRoot` backend.
Replica ID	The unique identifier of a replica that is set automatically in the corresponding Replication Domain configuration entry at each participating server. The replica ID identifies the source of each update in the replication topology.
Replica State	A list of the most recent change numbers of replication updates that have been applied to a replica. There can be at most one change number from each replica in the state. The replica state helps the Replication Server component to determine which updates the Replica has not received yet.
Replication	An automated background process that pushes directory server data changes to all other replicas.
Replication Changelog	A backend maintained by each replication server independently that records updates from each replica. This backend is distinct from the LDAP Changelog and the two should not be confused. The main distinction is as follows: ▪ The LDAP Changelog (i.e., the external changelog that clients can access) physically resides at `<server-root>/db/changelog`. ▪ The Replication Changelog Backend (i.e., the changelog that replication servers use) physically resides at `<server-root>/changelogDB` and is not accessible by clients and server extensions.
Replication Domain	The data configured for replication as defined by the base DN. Updates to entries at and below the base DN will be replicated.
Replication Replay	When a replica locally applies the update received via a replication server.
Replication Server	A component within the Directory Server process that is responsible for propagating directory server data changes to and from replicas. Each Directory Server instance participating in replication is also running a replication server.
Replication Server ID	The unique identifier of a replication server set automatically in its configuration object at each server in the replication topology. This identifier is used in the connection management code of the replication servers.
Replication Server State	A list of the most recent change numbers of replication updates that have been processed by a replication server. There can be at most one change number from each replica in the topology. The Replication Server State is used to determine which updates need to be sent to other replication servers. Similarly, the replica can use the Replication Server State to identify the set of updates to send to the replication server.
Sequence Number	A field in the change number that indicates the sequence of the client updates at a particular replica. For every update at the replica, the number is incremented. The initial value of the sequence number is 1. The number is stored as a 32-bit integer, but only positive values are used. The sequence number can roll over.

Term	Description
WAN Gateway	The designated replication server that assumes the WAN Gateway role within a collection of co-located replication servers (i.e., servers that are defined with identical location settings). Replication update messages between servers at different locations are routed through WAN gateways. The WAN Gateway role is assigned automatically by the protocol based on the server's WAN Gateway Priority setting. If the WAN Gateway server is down for any reason, the server with the next highest WAN Gateway Priority will dynamically assume the WAN Gateway role.
WAN Gateway Priority	The configuration setting that determines which replication server assumes the WAN Gateway role. The replication server with the lowest WAN Gateway Priority value in a location assumes the role of the WAN Gateway. The priority values can be set to 0 (never be a gateway), or any value from 1 (highest priority) to 10 (lowest priority).

Replication architecture

The major elements of replication in the PingDirectory Server are introduced in this section.

Eventual consistency

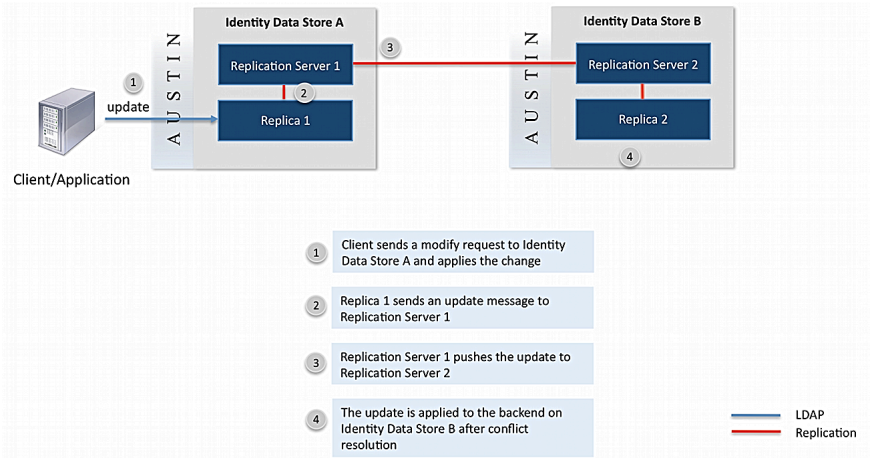
Replication is based on an eventual-consistency model, where updates that are propagated through a connected network of servers will eventually be consistent on all servers over a very short period of time. In a typical update operation, a client application updates an entry or group of entries on the PingDirectory Server with an ADD, DELETE, MODIFY, or MODIFY DN operation. After processing the operation, the Directory Server returns an LDAP response, while concurrently propagating the update to the other servers in the replicated topology. This concurrent processing model allows the client to continue submitting update requests without waiting for a replication completion response from the server. Alternatively, assured replication can be configured for specific write requests that requires local or global consistency, across datacenter locations, before a response is returned to the client. For more information, see *Configuring assured replication* on page 673.

To support this processing model, replication never locks the targeted entries at the other Directory Server instances before an update can be made locally. This means that the replicated Directory Servers may have an inconsistent view of the targeted entry for a very short period of time but will catch up as the propagated changes are applied. The eventual-consistency model also allows clients to complete update operations faster, since clients do not have to wait for replication to propagate the change. The rate of update operations remains the same no matter how many Directory Servers participate in replication.

Replicas and replication servers

Each Directory Server has an embedded replication server that is responsible for transmitting updates to other replication servers. There is a separate component, called a replica, for each replicating base DN, such as `cn=schema, dc=example,dc=com`. Each replica connects to the embedded replication server running within the Directory Server JVM process.

When a client application updates an entry on the Directory Server, the replica sends an update message to its embedded replication server. The replication server applies the change to the `replicationChangelog` backend repository and then sends an update message to the connected replication server on another directory server. The replication server on other directory servers then passes the change to the appropriate replica, which in turn applies the change to its backend after performing conflict resolution. This standard setup is seen in the figure below.



MY TITLE Replicas and Replication Servers

Authentication and authorization

The authentication in the Replication Protocol is based on public key cryptography using client certificate authentication via TLS. The certificate used for authentication is stored in the `ads-truststore` backend of the Directory Server. During replication setup, the command-line utility distributes public keys to all directory servers to establish trust between the Directory Servers and to enable client authentication via TLS.

The authorization model of replication is simple: once authenticated, the remote Directory Server is fully authorized to exchange replication messages with the local Directory Server. There is no other access control in place.

Logging

The access log messages in the Directory Server indicate if the update was received via replication and includes the corresponding change number. This allows the administrator to track which Directory Server the update originated from.

Replication deployment planning

The following should be considered before deploying in a production environment:

- **Minimizing Replication Conflicts**. Attention should be paid to the origin of client write requests to prevent a conflict. If two different clients attempt to create an entry with the same name, DN, at the same time against different servers, the possibility exists that both client requests will succeed. In this case, a conflict alert will be sent by the server. However, understand the client traffic pattern beforehand will minimize these occurrences. The Directory Server's Assured Replication feature and the PingDirectoryProxy Server can both assist in minimizing conflicts.
- **Replication Purge Delay**. Adjust the default one-day replication purge delay, consistently across all servers, to accommodate automatic catchup of changes when a server is offline for an extended period of time. The replication changes database, stored in `<server-root>/changelogDb`, grows larger as the replication purge delay is increased. A minimum value should be defined.

The rest of this section highlights other topics of consideration.

Location

In multi-site deployments, it is strongly recommended to configure the directory servers with location information using the **`dsconfig create-location`** command and **`dsconfig set-global-configuration-prop`** command. The Directory Server cannot determine LAN boundaries automatically, so incorrect location settings can result in undesired WAN communication. By default, replication also compresses all traffic between directory servers in different locations.

We recommend setting up the locations prior to enabling replication. The **`dsreplication enable`** command prompts for location information if you have forgotten to define the property.

User-defined LDAP

Directory Servers participating in replication are required to have a uniform user-defined schema. The **`dsreplication`** command-line utility sets up replication for the schema backend the first time replication is enabled to ensure that future schema changes are propagated to all directory servers.

Disk space

Replication increases the disk space required for the Directory Server. The Replication Changelog backend keeps changes from all directory servers for 24 hours by default. After this time period, also known as the purge delay, the backend is trimmed automatically.

In addition, within the `userRoot` and other local DN backends, attribute-level changes are recorded for a short period of time in the `ds-sync-hist` attribute of the entry targeted by an LDAP Modify operation. This attribute is used to resolve all conflicts resulting from LDAP Modify operations automatically.

The disk space impact of replication is highly dependent on the rate and size of changes in the replication topology, and the Replication Changelog purge delay.

Memory

Compared to a standalone directory server, replicated directory server instances require slightly more memory. All of the items discussed in the Disk Space section have an impact on the amount of memory the Directory Server is using. The additional replication overhead is typically less than 5%.

Time synchronization

Even though replication has a built-in mechanism to compensate for the potential clock skew between hosts, it is generally recommended to keep system clocks in sync within the deployment.

Communication ports

The replication server component in each directory server listens on a TCP/IP port for replication communication (the replication server port). This port, typically 8989, must be accessible from all directory servers participating in replication. The server-to-server communication channel is kept alive using a heartbeat, which occurs every 10 seconds. This traffic will prevent firewalls from closing connections prematurely.

The replication command-line utility (**`dsreplication`**) requires access to all directory servers participating in replication. This includes the LDAP or LDAPS port of the directory servers.

Keep these communication requirements in mind when configuring firewalls.

Hardware load balancers

Replication allows writes to be directed to any directory server in the topology. Distributing write operations in a round-robin fashion, however, may introduce conflicts. In particular, distributing a series of LDAP Add, Delete and Modify DN operations targeting the same DN in quick succession can result in conflicts that require manual intervention. The Assured Replication feature can help prevent conflicts created by the same client application.

If possible, consider using server affinity with the load balancer that either associates a client IP address or a client connection with a single directory server instance at a time. Also, consider using the PingDirectoryProxy Server for load balancing LDAP traffic. The Server Affinity feature in the Directory Proxy Server enables replication-friendly load balancing.

Directory Proxy Server

In addition, to facilitate replication-friendly load balancing, the PingDirectoryProxy Server should be considered in every replication deployment. The Directory Proxy Server can automatically adapt to

conditions in backend directory servers using health checks and route traffic accordingly. For example, traffic can be re-routed from directory servers with large backlog of replication updates.

Displaying the server information for a replication deployment

Steps

- Run the **`dsreplication status`** command with the `--displayServerTable` option.

```
$ bin/dsreplication status --displayServerTable
```

```
          --- Replication Status for dc=example,dc=com: ---
Server                          : Location : Priority (1) : Status
--------------------------------:----------:--------------:----------
austin-01.example.com:8989      : US       : 1 (*)        : Available
austin-02.example.com:8989      : US       : 2            : Available
london-01.example.com:8989      : UK       : 5            : Available
london-02.example.com:8989      : UK       : 5 (*)        : Available
sydney-01.example.com:8989      : AU       : 5 (*)        : Available
sydney-02.example.com:8989      : AU       : 5            : Available

[1] WAN Gateway Priority. WAN gateways are marked with a *. To minimize
WAN utilization, the server with the WAN Gateway role is the only server
in a location that exchanges updates with remote locations.

--- Replication Status for dc=example,dc=com: Enabled ---
Server                          : Location : Entries : Conflict Entries :
 Backlog : Recent Change Rate
--------------------------------:----------:---------:------------------:---------:---
austin-01.example.com:1389      : US       : 3478174 : 0                : 8
      : 333
austin-02.example.com:1389      : US       : 3478174 : 0                : 5
      : 345
london-01.example.com:1389      : UK       : 3478174 : 0                : 0
      : 349
london-02.example.com:1389      : UK       : 3478174 : 0                : 0
      : 5
sydney-01.example.com:1389      : AU       : 3478173 : 0                : 0
      : 350
sydney-02.example.com:1389      : AU       : 3478270 : 0                : 30
      : 332
```

Displaying all status information for a replication deployment

Steps

- Run the **`dsreplication status`** command with the `--showAll` option. You can also use the `--showAll` option together with the `--displayServerTable` option to see the server table information for the replication topology.

Enabling replication

Enabling replication between multiple Directory Servers means that any change within the replicated base DN is automatically propagated to all other Directory Servers in the topology. Configuration changes are not replicated, but can be through the use of Server Groups. Therefore, each Directory Server should be configured according to the deployment design.

Overview

To interface with the replication topology, the Directory Server provides a command-line utility, **`dsreplication`**, that must be used to manage and monitor replication.

The cluster name for an instance should be set to a unique name during install. The cluster-name controls a handle of configuration settings that are used cluster-wide; any changes to the configuration will always get pushed out to other members of the topology. If the cluster-name is set to a unique name for each server, any change to the configuration via the **dsconfig** command or the console needs to be done on every server in the topology. In a devops model, it is recommended that the cluster name be unique so that everything can be managed with server profiles. If you are doing an on-premise install, you may want to use a common cluster-name across the topology.

ⓘ **Note:** You should set a cluster name for the topology if you want cluster-wide settings to propagate across the topology in a single action.

Replication setup involves the following basic steps:

- **Set up the servers**. This is the basic installation steps to set up a Directory Server instance.
- **Import or restore data to one server**. After setting up the servers, at least one server should have the target data loaded through **import-ldif** or restore.
- **Enable replication between the servers**. Using the **dsreplication** tool, enable replication for each server to be included in the replication topology. The **dsreplication enable** subcommand should be run *N - 1* times for a topology of *N* servers. See *Command Line Interface* for more information.
- **Initialize data from source server to every server in the topology**. Run the **dsreplication initialize** subcommand for every target server that needs a copy of the data from the source server.
- **Verifying the replication topology**. Administrators can check the replication status after configuring the topology using the **dsreplication** status tool.

Command-line interface

Replication topologies are configured and maintained using the **dsreplication** command-line utility, which supports interactive and non-interactive modes. If you are running the server for the first time, we recommend using the **dsreplication** tool in interactive mode.

The **dsreplication** tool has the following format including some important subcommands listed in the section *The dsreplication Command-Line Utility*:

dsreplication `{subcommand} {connection parameters}`

The **dsreplication** tool keeps a history of invocations in the `logs/tools/dsreplication.history` file and keeps a log of up to 10 **dsreplication** sessions in the `logs/tools` directory.

What happens when you enable replication

The **dsreplication** `enable` subcommand is used to set up replication. The `enable` subcommand carries out the following functions:

- If it does not already exist, the global administrator user is created. The global administrator user has all the rights and privileges to update replication-related configuration objects. Most **dsreplication** subcommands require the global administrator.
- The server instances are registered in the `cn=topology,cn=config` tree. The registration includes basic host name, port information as well as the public key used during the replication authentication process.

  In case both servers are already participating in replication, the `cn=topology,cn=config` is merged to retain the server information from existing topologies.
- The embedded replication server is enabled. Servers already in replication will see their replication server configuration updated with the information of the new replication server.
- A replication domain is created for the requested base DNs. In case the first base DN is enabled, the replication domains for two additional base DNs are also enabled: `cn=topology,cn=config` and `cn=schema`.

- Initialization for the `cn=schema` base DN is executed. This will ensure that a uniform schema is present in the replication topology.
- Initialization must be performed for the enabled base DNs.

Initialization

Replica initialization transfers of a copy of the backend containing the replication domain to a target server. This should be performed after replication is enabled with the **`dsreplication initialize`** subcommand. There is no impact on the source server during this process.

ⓘ **Note:** When enabling or initializing servers, the **`--topologyFilePath`** option can be used with **`dsreplication`** to specify a file with a series of hosts and ports available in the topology that can be used as source servers. This option is used in place of specifying host 1, port1. When the hosts file is used for an enable or initialize action, the servers in the file are tried sequentially until the new server is successfully enabled or added. The rest of the servers in the file are ignored. This ensures that a host server is always available for replication. This file is generated with the **`manage-topology export`** command.

- **dsreplication initialize**. The recommended approach for replica initialization. The **`dsreplication initialize`** subcommand performs the most efficient copy of data needed to initialize one or more replicas on a target server. Any existing data on the target server replica will be lost and the backend containing the base DN will be taken offline on the target server during the initialization.
- **Binary Copy**. The database copy method involves copying database backup files from the source directory server to one or more target servers. The Directory Server provides tools necessary for backing up and restoring backends. Using **`server-root/bin/backup`**, create a backup of the backend containing the replicated base DN. The backup files then need to be transferred to the target server(s) and restored individually with **`server-root/bin/restore`**. There are additional considerations when using database copy as the means to initialize a target replica:
  - If encryption is enabled on the servers, then a database **`bin/encryption-settings export`** then **`bin/encryption-settings import`**must be performed on the `encryption-settings` backend.
  - When using database copy to initialize a server which has been offline longer than the replication purge delay, the database copy of the `replicationChanges`, `schema`, and `adminRoot` backends are required.

Replica generation ID

Each replica has a generation ID, which is an integer that summarizes the replica. It provides replication with a quick and simple means of determining if two replicas contain the same data. If they do contain the same data, they'll have the same generation ID.

When replication is operating correctly, all of the replicas for each replicated base DN will have the same generation ID. The generation ID is stored on each replica as the operational attribute **`ds-sync-generation-id`**. For example:

```
ldapsearch -b 'dc=example,dc=com' -s base '(&)' ds-sync-generation-id
 dn: dc=example,dc=com
 ds-sync-generation-id: 2058329333
```

When the server starts, or when replication is enabled, the generation ID is computed for each affected replica that does not already have a generation ID stored as **`ds-sync-generation-id`**. The following is used to calculate the generation ID:

- The total number of entries in the replica. This is referred to as "the count."
- The first 1000 entries in the replica are converted to normalized LDIF, which is referred to as "the LDIF." Normalized LDIF only includes attributes **`objectclass`**, **`sn`**, **`cn`** and **`ds-entry-unique-id`**, and uses OIDs in place of attribute names.

- The Adler-32 checksum is calculated with the string produced by concatenating the count and the LDIF as input. This Adler-32 checksum is the generation ID.
- The generation ID is stored on the base DN as **`ds-sync-generation-id`**. This is so that the ID does not need to be computed the next time the replica is loaded.

Deploying a basic replication topology

About this task

This section describes how to set up a two-server replication topology. The example uses the LDAP and replication server ports 1389 and 8989 respectively.

Replica Ports

Host Name	LDAP Port	Replication Port
server1.example.com	1389	8989
server2.example.com	1389	8989

Steps

1. Install the first directory server with 2000 sample entries.

```
$ ./setup --cli --acceptLicense --baseDN "dc=example,dc=com" --ldapPort 1389
 \ --rootUserPassword pass --sampleData 10000 --no-prompt
```

2. Install the second directory server either on a separate host or the same host as the first, but with a different LDAP port.

```
$ ./setup --cli --acceptLicense --baseDN "dc=example,dc=com" --ldapPort 1389
 \ --rootUserPassword pass --no-prompt
```

3. From the first server, run the **`bin/dsreplication`** command in interactive mode to configure a replication topology:

```
$ bin/dsreplication
```

4. From the Replication Main menu, select the Manage the topology option.

```
>>>> Replication Main Menu

What do you want to do?

    1) Display replication status
    2) Manage the topology (add and remove servers)
    3) Initialize replica data over the network
    4) Initialize replica data manually
    5) Replace existing data on all servers

    q) quit

Enter choice: 2
```

5. From the Manage Replication Topology menu, choose the Enable Replication option.

```
>>>> Manage Replication Topology

Select an operation for more information.

    1)  Enable Replication -- add or re-attach a server to the topology
    2)  Disable Replication -- permanently remove a running replica from the
 topology
```

```
    3)  Remove Defunct Server -- permanently remove an unavailable server
 from the
          topology
    4)  Cleanup Server -- remove replication artifacts from an offline,
 local server
          (allowing it to be re-added to a topology)

    b)  back
    q)  quit

Enter choice [b]: 1
```

6. On the Enable Replication menu, read the brief introduction on what will take place during the setup, and then, enter "c" to continue the enable process.
7. Next, enter the LDAP connection parameters for the first of the two replicas that you are configuring. First, enter the host name or IP address of the first server.
8. Next, enter the type of LDAP connection to the first server: 1) LDAP, 2) LDAP with SSL, or 3) LDAP with StartTLS.
9. Type the LDAP listener port for the first replica. If you are a root user, you will see port 389 as the default. Others will see port 1389.
10. Authenticate as a root DN, such as **`cn=Directory Manager`**. You will be prompted later in the process to set up a global administrator and password. The global administrator is the user ID that manages the replication topology group.
11. Repeat steps 7–10 for the second replica.
12. Next, the **`dsreplication`** tool checks for the base DN on both servers. In order to enable replication, data must be present on at least one of the servers. For this example, press Enter to select the default base DN, `dc=example,dc=com`.

```
Choose one or more available base DNs for which to enable replication:

    1)  dc=example,dc=com

    c)  cancel

Enter one or more choices separated by commas [1]:
```

> ⓘ **Note:** If you see the following message:
>
> ```
> There are no base DNs available to enable replication between the two
>  servers.
> ```
>
> In most cases, a base DN was not set up on one of the directory servers or the backend is disabled.

13. Next, the prompt asks if you want to set up entry balancing using the Directory Proxy Server. Press Enter to accept the default (no), since we are not setting up replication in an entry-balanced environment in this scenario. For more information, see the *PingDirectoryProxy Server Administration Guide*.

```
Do you plan to configure entry balancing using the Directory Proxy Server?
 (yes / no) [no]:
```

14. Next, enter the replication port for the first replica (default, 8989). The port must be free.
15. If the first server did not have a pre-defined location setting, **`dsreplication`** will prompt you to enter a location. Press Enter to accept the default (yes) to set up a Location for the first server. Enter the name of the server's location.

```
The first server has not been configured with a location.
Assigning a location to each server in the replication topology reduces
network traffic in multi-site deployments. Would you like to set the
```

```
location in the first server? (yes / no) [yes]

The location of the first server: Austin
```

16. Repeat the previous steps for the second directory server. Again, if you did not pre-define a location setting for the second server, you will be prompted to enter this information.
17. At this time, set up the Global Administrator user ID (default is "admin") and a password for this account. The Global Administrator user ID manages the directory servers used in the replication topology.

```
Specify the user ID of the global administrator account
that will be used to manage the Ping Identity
                  Directory Server
instances to be replicated [admin]:

Password for the global administrator:
Confirm Password:
```

18. Return to the Replication Main Menu and enter the number corresponding to initializing data over the network.
19. On the Initialize Replica Data over the Network menu, select Initialize to initialize data on a single server, and then enter `c` to continue.
20. Next, specify a server in the replication topology. For this example, enter the host name or IP address, LDAP connection type, LDAP port, Global Admin user ID and password of the first server.
21. Next, select the source server that is hosting the data to which the target server will be initialized. For this example, select the first server, since the sample dataset has been loaded onto this server.
22. Next, select the base DN that will be initialized. In most cases, the base DN for the root suffix will be replicated. In this example, `dc=example,dc=com`.
23. Next, select the second server in this example that will have its data initialized, and then enter the Global Admin user ID and password for the target server. Any data present on the target server will be over-written.
24. Press Enter to confirm that you want to initialize data on the target server. When completed, you should see "Base DN initialized successfully."

```
Initializing the contents of a base DN removes all the existing contents of
that base DN. Do you want to remove the contents of the selected base DNs on
 server
server2.example.com:1389 and replace them with the contents of server
server1.example.com:1389? (yes / no) [yes]:
```

25. On the Initialize Replica Data Over Network menu, enter `b` to back out one level to the main menu. Then, on the Replication Main menu, enter the number to view the replication status.

```
     --- Replication Servers: dc=example,dc=com ---
Server                  : Location : Conflict Entries : Backlog : Recent
 Change Rate
------------------------:----------:------------------:---------:--------------------
ds1 (example.com:1389)   : austin   : 0                : 0       : 0
ds2 (example.com:1389)   : austin   : 0                : 0       : 0
```

Example deployment with non-interactive dsreplication

This example will create a four-server topology spanning two data centers. The four servers are already installed and have locations Austin and Budepest defined.

ⓘ **Note:** When enabling or initializing servers, the **`--topologyFilePath`** option can be used with **`dsreplication`** to specify a file with a series of hosts and ports available in the topology that can be used as source servers. This option is used in place of specifying host 1, port1. When the hosts file is used for an enable or initialize action, the servers in the file are tried sequentially until the new server is

successfully enabled or added. The rest of the servers in the file are ignored. This ensures that a host server is always available for replication. This file is generated with the **manage-topology export** command.

Deploying with non-interactive dsreplication

Steps

1. Generate the sample data using the **make-ldif** tool on the first server.

```
$ bin/make-ldif --templateFile config/MakeLDIF/example-10K.template \
  --ldifFile ldif/10K.ldif
```

2. Select a server from which to import data, and to be the source for future initialization to other servers. Stop this server, import the sample LDIF and start again, or perform a task-based **import-ldif** with the connection options.

```
$ bin/stop-server
$ bin/import-ldif --backendID userRoot --ldifFile ldif/10K.ldif
$ bin/start-server
```

3. Enable replication by choosing a specific server and running **dsreplication enable** three times. For the first invocation, create the replication topology administrator with the name **admin**.

```
$ bin/dsreplication enable --host1 austin01.example.com --port1 1389 \
    --bindDN1 "cn=Directory Manager" --bindPassword1 password \
    --replicationPort1 8989 --host2 austin02.example.com --port2 1389 \
    --bindDN2 "cn=Directory Manager" --bindPassword2 password \
    --replicationPort2 8989 --baseDN dc=example,dc=com --adminUID admin \
    --adminPassword password --no-prompt

$ bin/dsreplication enable --host1 austin01.example.com --port1 1389 \
    --bindDN1 "cn=Directory Manager" --bindPassword1 password \
    --replicationPort1 8989 --host2 budapest01.example.com --port2 1389 \
    --bindDN2 "cn=Directory Manager" --bindPassword2 password \
    --replicationPort2 8989 --baseDN dc=example,dc=com --adminUID admin \
    --adminPassword password --no-prompt

$ bin/dsreplication enable --host1 austin01.example.com --port1 1389 \
    --bindDN1 "cn=Directory Manager" --bindPassword1 password \
    --replicationPort1 8989 --host2 budapest02.example.com --port2 1389 \
    --bindDN2 "cn=Directory Manager" --bindPassword2 password \
    --replicationPort2 8989 --baseDN dc=example,dc=com --adminUID admin \
    --adminPassword password --no-prompt
```

4. Initialize the other servers (the dc=example,dc=com replicas) from the server that had the data imported with **import-ldif**. To minimize the WAN transfers, initialize **budapest02** from **budapest01**.

```
$ bin/dsreplication initialize --hostSource austin01.example.com --
portSource 1389 \
  --hostDestination austin02.example.com --portDestination 1389 \
  --adminUID admin --adminPassword password --baseDN dc=example,dc=com --no-
prompt

$ bin/dsreplication initialize --hostSource austin01.example.com --
portSource 1389 \
  --hostDestination budapest01.example.com --portDestination 1389 \
  --adminUID admin --adminPassword password --baseDN dc=example,dc=com --no-
prompt

$ bin/dsreplication initialize --hostSource budapest01.example.com --
portSource 1389 \
  --hostDestination budapest02.example.com --portDestination 1389 \
```

```
  --adminUID admin --adminPassword password --baseDN dc=example,dc=com --no-
prompt
```

5. View the state of replication.

```
$ bin/dsreplication status --adminPassword password --no-prompt --
displayServerTable --showAll
```

Using dsreplication with SASL GSSAPI (Kerberos)

Before you begin
This example procedure assumes that you have configured SASL GSSAPI on all servers in the replication topology and that they are working properly.

About this task
The Directory Server's utilities all support SASL GSSAPI options for systems using Kerberos as its main authentication mechanism. The following procedure shows how to use **`dsreplication`** with SASL GSSAP to set up a new **`replication.admin`** identity, while enabling replication on a server. The following are important points about the configuration:

- A separate Kerberos identity is required to manage replication. Existing Kerberos credentials can be used to interact with the server when enabling replication and creating the new identity.
- The new identity, such as replication.admin, must not exist as the cn or uid value under any public base DN.

Steps

1. Set the LDAP Connection Handler to explicitly listen on the server's host name address.

```
$ bin/dsconfig set-connection-handler-prop \
  --handler-name "LDAP Connection Handler" \
  --remove listen-address:0.0.0.0 --add listen-address:host.example.com
```

2. Update the identity mapper to have `cn=topology,cn=config` included in the list of base DNs and to add the `cn` attribute to match attributes. This step is required to map the admin account to the Kerberos realm.

```
$ bin/dsconfig set-identity-mapper-prop \
  --mapper-name "Regular Expression" \
  --add match-attribute:cn \
  --set "match-base-dn:cn=topology,cn=config" \
  --set match-base-dn:dc=example,dc=com
```

3. Invoke replication enable, authenticating as the existing kerberos `authid`. Note that no bind DNs and passwords are required to authenticate because we are using SASL binding. However, the new replication admin user requires a password at creation time, so we recommend using a strong random password. Once SASL is working, you will no longer have to provide this random password. Also note that we are forcing the use of the ticket cache, so make sure you have properly authenticated as `ds.admin` from your local host and the ticket is not expired in the cache.

```
$ kinit -p ds.admin
$ bin/dsreplication enable \
  --host1 server1.example.com --port1 1389 --replicationPort1 1989 \
  --host2 server2.example.com --port2 2389 --replicationPort2 2989 \
  --baseDN dc=example,dc=com \
  --adminUID replication.admin --adminPassword strongPassword \
  --saslOption1 mech=gssapi --saslOption1 authid=ds.admin@EXAMPLE.COM \
  --saslOption1 useTicketCache=true --saslOption1 requireCache=true \
  --saslOption2 mech=gssapi --saslOption2 authid=ds.admin@EXAMPLE.COM \
  --saslOption2 useTicketCache=true --saslOption2 requireCache=true
```

4. Use **`dsreplication initialize`** to initialize data on remote server.

```
$ kinit -p replication.admin
$ bin/dsreplication initialize \
  --hostSource server1.example.com --portSource 1389 \
  --hostDestination server2.example.com --portDestination 2389 \
  --baseDN dc=example,dc=com \
  --saslOption mech=GSSAPI \
  --saslOption authID=replication.admin@EXAMPLE.COM  \
  --no-prompt
```

5. The temporary **`userPassword`** can now be deleted from the **`replication.admin`** entry. Create a file called **`remove-password.ldif`** with these contents:

```
dn: cn=replication.admin,cn=Administrators,cn=topology,cn=config
changetype: modify
delete: userPassword
```

6. Apply the modifications using **`ldapmodify`**:

```
$ ./ldapmodify --filename remove-password.ldif -o mech=GSSAPI
-o authid=replication.admin@example.com \
  --saslOption useTicketCache=true \
  --hostname host.example.com --port 1389 \
  --noPropertiesFile
```

7. Check the topology's status by running **`dsreplication status`**. The `--saslOption useTicketCache=true` and `--saslOption requireCache=true` properties, instead of providing a password, for all **`dsreplication`** commands after properly creating the admin accounts and mappers.

```
$ bin/dsreplication status \
  --saslOption mech=gssapi \
  --saslOption authid=replication.admin@EXAMPLE.COM \
  --saslOption useTicketCache=true --saslOption requireCache=true \
  --hostname host.example.com --port 1389 \
  --no-prompt
```

Configuring assured replication

The PingDirectory Server's replication mechanism is based on the eventual-consistency model, which is a loosely-connected topology that propagates updates to other servers without waiting for their corresponding replication response messages. As a result, there is a small window of time where updates are not all present on the replicas as the changes are replicating through the system. There are, however, deployments that require *immediate* access to replicated data. In such cases, administrators can configure *Assured Replication*, which ensures that replication has completed *before* the update response is returned to the client.

From the LDAP client's perspective, assured replication has no bearing on the result code of the operation, just on the time in which it takes to receive the response for those requests in which replication assurance is applied. Detailed information regarding assurance processing is available to an LDAP client with awareness of the assured replication control.

The assured replication mechanism takes advantage of server location to distinguish between local and remote servers to allow different policies to apply. For example, a common assurance approach is to respond to a client update after all servers in the same location have applied the update, and one or more servers in remote locations have received the update. In addition, the level of assurance applied to each operation can be explicitly requested by the client and/or specified by the server configuration using the Replication Assurance Policy.

Assured replication is supported by client requests directly to PingDirectory Server and/or through a PingDirectoryProxy Server.

About the Replication Assurance Policy

Assured replication uses a *Replication Assurance Policy* to define the properties needed to ensure that replication has satisfactorily completed before the update response is returned to the client. Multiple policies can be defined but only one policy is matched with a client update request. Each policy contains an evaluation order index which, together with an optional request and connection criteria, provides flexibility in matching a policy to request.

The Replication Assurance Policy defines local and remote assurance *levels*. A *level* defines how rigorous the policy should be when waiting for propagation of updates, while *local* and *remote* distinguish servers in the same location versus servers in remote locations. Although optional, it is recommended that request or connection criteria be associated with a policy to apply replication assurance appropriately.

The Directory Server contains a Default Replication Assurance Policy that is enabled but has no assurance levels assigned. In addition to using the Default Replication Assurance Policy, any number of policies can be created and enabled. When a client request is received, the server iterates through the list of enabled policies according to each policy's *evaluation-order-index* property. A smaller *evaluation-order-index* value (e.g., 1) has precedence over policies with larger *evaluation-order-index* values (e.g., 2, 3, 4, etc.). The *evaluation-order-index* values do not need to be contiguous. The first policy that matches a request is associated with the operation.

The Replication Assurance Policy, which is defined on the PingDirectory Server and not on the PingDirectoryProxy Server, has the following properties:

- **evaluation-order-index**. Determines the evaluation order (the smaller value has precedence) among multiple Replication Assurance Policies that match against an operation.
- **local-level**. Specifies the assurance level used to replicate to local servers. A local server is defined as a server with the same `location` property value as set in the global configuration. The `local-level` property must be set to an assurance level as strict as the `remote-level` property. For example, if the `remote-level` is set to "`processed-all-remote-locations`," then the `local-level` property must be "`processed-all-servers`."
  - **None**. Replication to any local server is not assured.
  - **received-any-server**. At least one available local server must receive a replication update before a response is sent to the client.
  - **processed-all-servers**. All available local servers must complete replay of the replication update before the response is sent to the client. If a singular server is enabled, or only one server is available for a particular location, `processed-all-servers` will return a value of `false`.
- **remote-level**. Specifies the assurance level used to replicate to remote servers. A remote server is defined as a server that has a different `location` property value as set in the global configuration.
  - **None**. Replication to any remote server is not assured.
  - **received-any-remote-location**. At least one server at any available remote location must receive a replication update before a response is sent to the client.
  - **received-all-remote-locations**. All available remote servers must receive a replication update before the response is sent to the client.
  - **processed-all-remote-servers**. All available servers at all locations must complete replay of the replication update before the response is sent to the client. If a single server is enabled, or only one server is available for a particular location, `processed-all-remote-servers` will return a value of `false`.
- **timeout**. Specifies the maximum length of time to wait for the replication assurance requirements to be met before timeout and replying to the client.
- **connection-criteria**. Specifies connection criteria used to indicate which operations from clients matching this criteria use this policy. If both connection criteria and request criteria are specified for a policy, then both must match an operation for the policy to be assigned.
- **request-criteria**. Specifies request criteria used to indicate which operations from clients matching this criteria use this policy. If both connection criteria and a request criteria are specified for a policy, then both must match an operation for the policy to be assigned.

Servers in a replication topology are not required to share a homogeneous set of policies; you can configure the Replication Assurance Policies differently on the replicas in a topology. For example, if you configure server A to match add operations to a `processed-all-servers` assurance level, and server B to match add operations to a local `received-any-server` level, then add operations received on server A will have the assurance level of `processed-all-servers` and add operations received on server B will have the assurance level of `received-any-server`.

For more detailed information, see the *PingDirectory Server Configuration Reference Guide*.

Points about assured replication

The following are some points when implementing Assured Replication:

- **Client Controls**. The client may optionally include an assured replication request control with each operation. This control allows the client to specify bounds on assurance levels and/or override the timeout assigned by the matching replication assurance policy for the associated operation. The server always honors these request controls. See *About the Assured Replication Controls* for more information.
- **Directory Proxy Server**. Replication assurance policies are not supported on the PingDirectoryProxy Server. Replication client controls are passed through to the underlying Directory Server backend.
- **Schema backend Replication**. The schema backend is not supported by Assured Replication. Replication assurance policies that include criteria to match this backend will be rejected.
- **Backward Compatibility**. Server versions that support assured replication are backwards-compatible with prior versions that do not support assured replication.
- **WAN-Friendly Replication**. Assured replication functions independently from WAN-Friendly Replication and the notion of WAN Gateways.
- **Global Configuration Properties**. The Directory Server provides two configurable global configuration properties that determine the timing of the assurance source and maximum number of replication backup updates to be recognized as an available source.
  - **replication-assurance-source-timeout-suspend-duration**. Specifies the amount of time a replication assurance source will be suspended from assurance requirements if it experiences an assurance timeout. While suspended, the source will be excluded from assurance requirements for all operations originating on this Directory Server. This avoids the situation of repeated timeouts caused by degraded or offline servers. Default is 10 seconds.
  - **replication-assurance-source-backlog-fast-start-threshold**. Specifies the maximum number of replication backlog updates a replication assurance source can have and be immediately recognized as an available source. If a source connects to this server with more than the configured threshold backlog updates, it will be excluded from assurance requirements for all operations originating from this Directory Server until it completes at least one assurance successfully (i.e. this Directory Server receives an update acknowledgement message from it within the timeout window). Default is 1000.

Configuring assured replication

About this task

This example illustrates configuring a variety of assured replication policies. In practice it's common for all servers to have the same policy. The following example assumes that three servers are configured on localhost, on ports 1389, 2389 and 3389. Note that each server has a default Replication Assurance Policy with no assurance levels set.

Steps

1. On server 1, use **`dsconfig`** to create request criteria for add operations. This request criteria will be used to match any add operation with the Replication Assurance Policy that will be configured in the next step.

```
$ bin/dsconfig create-request-criteria \
```

```
  --criteria-name Adds \
  --type simple \
  --set operation-type:add
```

2. On server 1, set up the Replication Assurance Policy to make all add operations assured with a level of `processed-all-servers`, which indicates that all local servers in the topology must complete replay of the replication update before the response is sent to the client. Specify the Adds request criteria configured in the previous step.

```
$ bin/dsconfig create-replication-assurance-policy \
  --policy-name "Adds Processed All Locally" \
  --set evaluation-order-index:1 \
  --set local-level:processed-all-servers \
  --set "timeout:500ms" \
  --set request-criteria:Adds
```

3. On server 1, repeat the previous two steps for modify operations. The Replication Assurance Policy "Mods Received Any Locally" ensures that at least one available local server must receive a replication modify update before a response is sent to the client.

```
$ bin/dsconfig create-request-criteria \
  --criteria-name Mods \
  --type simple \
  --set operation-type:modify

$ bin/dsconfig create-replication-assurance-policy \
  --policy-name "Mods Received Any Locally" \
  --set evaluation-order-index:2 \
  --set local-level:received-any-server \
  --set "timeout:500ms" \
  --set request-criteria:Mods
```

4. On server 2, repeat steps 1-3 to set up the Adds and Mods request criteria and its respective Replication Assurance Policy.

```
$ bin/dsconfig create-request-criteria \
  --criteria-name Adds \
  --type simple \
  --set operation-type:add

$ bin/dsconfig create-request-criteria \
  --criteria-name Mods \
  --type simple \
  --set operation-type:modify

$ bin/dsconfig create-replication-assurance-policy \
  --policy-name "Adds Received Any Locally" \
  --set evaluation-order-index:1 \
  --set local-level:received-any-server \
  --set "timeout:500ms" \
  --set request-criteria:Adds

$ bin/dsconfig create-replication-assurance-policy \
  --policy-name "Mods Processed All Locally" \
  --set evaluation-order-index:2 \
  --set local-level:processed-all-servers \
  --set "timeout:500ms" \
  --set request-criteria:Mods
```

5. Leave server 3 with the default Replication Assurance Policy configured with no assurance levels or criteria. In practice it is common for all servers to have the same assurance levels or criteria.

6. On server 1, list the policies on the server using the **dsconfig** command to confirm that they exist on the server.

```
$ bin/dsconfig list-replication-assurance-policies
```

```
Replication Assurance Policy         : Type    : enabled : evaluation-order-index : local-level           : remote-level
-------------------------------------:---------:---------:-------------------------:-----------------------:-------------
Adds Processed All Locally           : generic : true    : 1                       : processed-all-servers : none
Mods Received Any Locally            : generic : true    : 2                       : received-any-server   : none
Default Replication Assurance Policy : generic : true    : 9999                    : none                  : none
```

7. Repeat the previous step on server 2 and server 3. Server 3 should only show the Default Replication Assurance Policy.
8. Check the Replication Assurance counters on all servers before any add or modify operation using `ldapsearch`. They should be set to zero. These counters are on the replica server, which is where the policy is matched and assigned. On server 1, run the following command:

```
$ bin/ldapsearch --baseDN "cn=Replica dc_example_dc_com,cn=monitor" \
  "(objectclass=*)" | grep replication-assurance

replication-assurance-local-completed-normally: 0
replication-assurance-local-completed-abnormally: 0
replication-assurance-local-completed-with-timeout: 0
replication-assurance-local-completed-with-shutdown: 0
replication-assurance-local-completed-with-unavailable-server: 0
replication-assurance-remote-completed-normally: 0
replication-assurance-remote-completed-abnormally: 0
replication-assurance-remote-completed-with-timeout: 0
replication-assurance-remote-completed-with-shutdown: 0
replication-assurance-remote-completed-with-unavailable-server: 0
```

9. Check the Replication Summary table on all of the servers. For example, on server 1, run the following command:

```
$ bin/ldapsearch --baseDN "cn=Replication Summary
 dc_example_dc_com,cn=monitor" \
  "(objectclass=*)" | grep replication-assurance

replication-assurance-submitted-operations: 0
replication-assurance-local-completed-normally: 0
replication-assurance-local-completed-abnormally: 0
replication-assurance-local-completed-with-timeout: 0
replication-assurance-local-completed-with-shutdown: 0
replication-assurance-local-completed-with-unavailable-server: 0
replication-assurance-remote-completed-normally: 0
replication-assurance-remote-completed-abnormally: 0
replication-assurance-remote-completed-with-timeout: 0
replication-assurance-remote-completed-with-shutdown: 0
replication-assurance-remote-completed-with-unavailable-server: 0
```

10. Add an entry to the server 1 Directory Server. Check that the counters matched the newly added entry to the "Adds Processed All Locally" Policy and that it completed assured.

```
$ bin/ldapmodify --filename add-user.ldif --defaultAdd

$ bin/ldapsearch --baseDN "cn=Replica dc_example_dc_com,cn=monitor" \
  "(objectclass=*)" | grep replication-assurance

replication-assurance-submitted-operations: 1
replication-assurance-local-completed-normally: 1
replication-assurance-local-completed-abnormally: 0
replication-assurance-local-completed-with-timeout: 0
replication-assurance-local-completed-with-shutdown: 0
replication-assurance-local-completed-with-unavailable-server: 0
replication-assurance-remote-completed-normally: 0
```

```
replication-assurance-remote-completed-abnormally: 0
replication-assurance-remote-completed-with-timeout: 0
replication-assurance-remote-completed-with-shutdown: 0
replication-assurance-remote-completed-with-unavailable-server: 0
replication-assurance-policy-matches: Adds Processed All Locally: 1
replication-assurance-policy-matches: Default Replication Assurance Policy:
 0
replication-assurance-policy-matches: Mods Received Any Locally: 0
replication-assurance-local-level-uses: processed-all-servers: 1
replication-assurance-remote-level-uses: none: 1

$ bin/ldapsearch --baseDN "cn=Replication Summary
 dc_example_dc_com,cn=monitor" \
  "(objectclass=*)" | grep replication-assurance

replication-assurance-submitted-operations: 1
replication-assurance-local-completed-normally: 1
replication-assurance-local-completed-abnormally: 0
replication-assurance-local-completed-with-timeout: 0
replication-assurance-local-completed-with-shutdown: 0
replication-assurance-local-completed-with-unavailable-server: 0
replication-assurance-remote-completed-normally: 0
replication-assurance-remote-completed-abnormally: 0
replication-assurance-remote-completed-with-timeout: 0
replication-assurance-remote-completed-with-shutdown: 0
replication-assurance-remote-completed-with-unavailable-server: 0
```

11. Perform a modify of an entry under dc=example,dc=com on server 1. Check that the counters matched the modify operation to the "Mods Processed All Locally" policy and that the operations completed assured.

```
$ bin/ldapsearch --baseDN "cn=Replica dc_example_dc_com,cn=monitor" \
  "(objectclass=*)" | grep replication-assurance

replication-assurance-submitted-operations: 2
replication-assurance-local-completed-normally: 2
replication-assurance-local-completed-abnormally: 0
replication-assurance-local-completed-with-timeout: 0
replication-assurance-local-completed-with-shutdown: 0
replication-assurance-local-completed-with-unavailable-server: 0
replication-assurance-remote-completed-normally: 0
replication-assurance-remote-completed-abnormally: 0
replication-assurance-remote-completed-with-timeout: 0
replication-assurance-remote-completed-with-shutdown: 0
replication-assurance-remote-completed-with-unavailable-server: 0
replication-assurance-policy-matches: Adds Processed All Locally: 1
replication-assurance-policy-matches: Default Replication Assurance Policy:
 0
replication-assurance-policy-matches: Mods Received Any Locally: 1
replication-assurance-local-level-uses: processed-all-servers: 1
replication-assurance-local-level-uses: received-any-server: 1
replication-assurance-remote-level-uses: none: 2

$ bin/ldapsearch --baseDN "cn=Replication Summary
 dc_example_dc_com,cn=monitor" \
  "(objectclass=*)" | grep replication-assurance

replication-assurance-submitted-operations: 2
replication-assurance-local-completed-normally: 2
replication-assurance-local-completed-abnormally: 0
replication-assurance-local-completed-with-timeout: 0
replication-assurance-local-completed-with-shutdown: 0
replication-assurance-local-completed-with-unavailable-server: 0
replication-assurance-remote-completed-normally: 0
```

```
replication-assurance-remote-completed-abnormally: 0
replication-assurance-remote-completed-with-timeout: 0
replication-assurance-remote-completed-with-shutdown: 0
replication-assurance-remote-completed-with-unavailable-server: 0
```

You have successfully configured Assured Replication.

About the assured replication controls

The LDAP SDK for Java provides an implementation of an LDAP control that can be included in add, bind, modify, modify DN, and certain extended requests to indicate the level of replication assurance desired for the associated operation. The OID for this control is 1.3.6.1.4.1.30221.2.5.28, and may have a criticality of either TRUE or FALSE.

For specific details, see the LDAP SDK javadoc for the `AssuredReplicationRequestControl` class.

Managing the topology

The following sections describe common topology management operations.

ⓘ **Note:** When enabling or disabling replication within a topology that contains multiple product versions, the tool must be run from the server root location of a member of the topology that has the oldest product version.

Adding a server to the topology

About this task
The following steps assume an existing directory server topology. The commands are identical for initial enable between two servers, where one server contains data for the replication domain stored in the **`userRoot`** backend. If database encryption is being used on the servers in the topology, it is important that the server being initialized has a copy of the `encryption-settings` backend from the source server.

Steps

1. A majority of servers (more than 50%) in the topology and the new server, should be online.
2. Enable replication for the base DN, or base DNs, using an existing server as **`host1`** and the new server as **`host2`**.

```
$ bin/dsreplication enable \
  --host1 austin01.example.com --port1 1389 \
  --bindDN1 "cn=Directory Manager" --bindPassword1 password \
  --replicationPort1 8989 --host2 austin03.example.com --port2 1389 \
  --bindDN2 "cn=Directory Manager" --bindPassword2 password \
  --replicationPort2 8989 --baseDN dc=example,dc=com --adminUID admin \
  --adminPassword password --no-prompt
```

3. Optionally, compare the configurations between the two hosts used in the **`dsreplication enable`** command. Make sure settings are consistent across the topology and are also consistent with the new system:

```
$ bin/config-diff --sourceLocal \
  --targetHost austin03.example.com \
  --targetBindDN "cn=directory manager" \
  --targetBindPassword pass --targetPort 1389
```

Disabling replication and removing a server from the topology

When removing a server from the topology, the remaining servers need to be made aware of the change. If the server to remove is online, only one invocation of **`dsreplication disable`** is necessary. If the server to remove is offline, the following steps are required:

1. Run **`remove-defunct-server`** from another server in the topology.
2. Run **`remove-defunct-server`** on the offline server to removed.

Similar to the **`enable`** command, more that 50% of servers not being removed from the topology must be online during the process.

If additional servers are offline and cannot be online while removing the server, you must distinguish between offline servers that are offline permanently and those that are offline temporarily. If servers are offline permanently, use **`remove-defunct-server`** to remove them. If servers are offline temporarily, they update automatically after they are online. The following examples show the steps in more detail:

- **Removing a server that is still online**. The **`dsreplication disable`** command can be run from any server, but a majority of servers in the topology need to be online.

  ```
  $ bin/dsreplication disable --hostname austin03.example.com --port 1389 \
    --baseDN dc=example,dc=com --adminUID admin --adminPassword password \
    --no-prompt
  ```

- **Removing a server that is offline**.The **`remove-defunct-server`** tool can be run against any server not being removed from the topology. More than 50% of servers in the topology must be online. The **`remove-defunct-server`** tool can be issued after setting the JVM property **`"com.unboundid.connectionutils.LdapResponseTimeoutMillis"`** to change the default ten-minute timeout for each server to take out of rotation. If multiple servers need to be removed, this approach speeds up the process.

  ```
  $ bin/remove-defunct-server --serverInstanceName austin01 \
    --bindDN "cn=Directory Manager" --bindPassword password
  ```

  To remove any topology references, run the **`remove-defunct-server`** tool on each server that is removed from the topology, as follows:

  ```
  $ bin/remove-defunct-server --no-prompt
  ```

The option **`--ignore-online`** removes an online server cleanly from the topology.

Replacing the data for a replicating domain

In the rare event that the data for the entire replication domain (such as the backend) needs to be replaced, perform the following steps:

Replacing the data

Steps

1. With all servers online, the **`dsreplication pre-external-initialization`** command must be run once against any server in the topology. This stops replication for the domain. No writes are made by clients to any of the servers.

   ```
   $ bin/dsreplication pre-external-initialization --hostname
    austin01.example.com \
     --port 1389 --baseDN dc=example,dc=com --adminUID admin \
     --adminPassword password --no-prompt
   ```

2. Using **`import-ldif`**, replace the data for **`dc=example,dc=com`**. Make sure that the input LDIF is free of any replication attributes by using the **`--excludeReplication`** option. The **`--overwriteExistingEntries`** option is necessary to overwrite the existing data for the domain. For example, to perform the **`import-ldif`** with the server offline:

   ```
   $ bin/import-ldif --ldifFile new-data.ldif --backendID userRoot --
   excludeReplication --overwriteExistingEntries
   ```

3. Initialize the other servers in the topology with **`dsreplication initialize`**, using the server which has the new data as the source host. For example:

```
$ bin/dsreplication initialize --hostSource austin01.example.com --
portSource 1389 \
   --hostDestination budapest01.example.com --portDestination 1389 \
   --adminUID admin --adminPassword password --baseDN dc=example,dc=com \
   --no-prompt
```

4. Run **`dsreplication post-external-initialization`** once from any server in the topology. All servers in the topology must be online:

```
$ bin/dsreplication post-external-initialization --hostname
 austin01.example.com \
   --port 1389 --baseDN dc=example,dc=com --adminUID admin \
   --adminPassword password --no-prompt
```

Advanced configuration

The following sections are advanced configuration procedures that may be appropriate for your company's deployment.

Changing the replicationChanges DB Location

About this task

You can change the `replicationChanges` DB location if on-disk space issues arise. The replication changelog database can live outside `<server-root>` and be placed in another location on the file system. In that case, you must specify the absolute path of the replication changelog directory.

Steps

1. Use **`dsconfig`** to change the database location for the replication changelog, which by default is at `<server-root>/changelogDb`. The following command sets the replication changelog backend to `<server-root>/data/directory/changelogDB` . Remember to include the LDAP connection parameters (host name, port, bindDN, bindPassword).

```
$ bin/dsconfig set-replication-server-prop \
  --provider-name "Multimaster Synchronization" \
  --set "replication-db-directory:/data/directory/changelogDb" \
  --bindDN "cn=Directory Manager" --bindPassword secret --no-prompt
```

2. Stop the server, move the DB files, and then restart the server.

```
$ bin/stop-server
$ mv changelogDb /data/directory
$ bin/start-server
```

Modifying the replication purge delay

About this task

The replication purge delay specifies the period after which the directory server purges changes on the replication server database. Any change that is older than the purge delay is removed from the replication server database regardless of whether the change has been applied.

Currently, the PingDirectory Server sets the default purge delay to one day. Administrators can change the default purge delay using the **`dsconfig`** tool. To ensure proper replication processing, you must have the same purge delay value set for all replication servers in the topology.

Steps

- Use **dsconfig** to change the purge delay. The property accepts time units of seconds (s), minutes (m), hours (h), days (d), or weeks (w). The following command is entered on the command line and changes the purge delay from the default one day to two days.

  In **dsconfig** interactive mode, open the **Advanced objects** menu. Select **Replication Server**. Select the replication synchronization provider, and then select the option to change the replication purge delay.

```
$ bin/dsconfig set-replication-server-prop \
  --provider-name "Multimaster Synchronization" \
  --set "replication-purge-delay:2 d"
```

Configuring a single listener-address for the replication server

About this task

By default, the replication server binds the listening ports to all available interfaces of the machine. To bind the listener to a specific address, the address must be the host name provided when replication is enabled and the `listen-on-all-addresses` property must be set to `FALSE`.

The replication server's configuration entry already stores a host name for itself so that it can resolve the address and specify it during the socket bind. If the server information is missing from the system, an error message will be generated with instructions on specific address binding. You can use the **dsconfig** tool to change the value of the `listen-on-all-addresses` property from TRUE (default) to FALSE.

To configure a replication server to listen on a single address:

Steps

1. Create a new directory serverinstance with replication enabled on port 8989.
2. Run **netstat** to see the ports bound for listening on port 8989. Notice that *.8989 means that it is listening on all addresses.

```
$ netstat -an | grep LISTEN | grep 8989
```

3. Run **dsconfig** to disable listening on all addresses for the replication server.

```
$ bin/dsconfig set-replication-server-prop \
  --provider-name "Multimaster Synchronization" \
  --set listen-on-all-addresses:false
```

4. Run **netstat** again to see the ports bound for listening on port 8989. Notice that `<address>.8989` (for example, 10.8.1.211.8989) means that it is listening on the one address.

Monitoring replication

Replication in the PingDirectory Server can be monitored the following ways:

- The **dsreplication** `status` subcommand displays basic information about the replicated base DNs, the number of entries replicated as well as the approximate size of replication backlogs at each Directory Server.
- The more detailed information about the state of replication can be obtained via the information exposed in its monitoring framework under `cn=monitor`. Administrators can monitor their replication topologies using several tools and protocols: the PingDataMetrics Server, SNMP, LDAP, JMX, or through the Administrative Console. See Managing Logging and Monitoring.
- The Periodic Stats Logger plugin allows collecting replication statistics for profiling server performance. For more information, see Profiling Server Performance Using the Periodic Stats Logger.

Monitoring replication using cn=monitor

The `cn=monitor` branch has a number of entries that store the replication state of a topology.

- **Direct LDAP Server <baseDN> <host name:port> <serverID>**. Defines an LDAP server that is directly connected to the replication server that you are querying. The information in this entry applies to the replication server local to the `cn=monitor` entry. For detailed information, see *Summary of the Direct LDAP Monitor Information*.
- **Indirect LDAP Server <baseDN> <serverID>**. Defines an LDAP server that is connected to another replication server in the topology. While this server is connected to the same topology, it is not connected to the replication server being queried. For detailed information, see *Summary of the Indirect LDAP Server Monitor Information*.
- **Remote Repl Server <baseDN> <host name:port> <serverID>**. Defines a remote replication server that is connected to the local replication server. Information in this entry is in respect to the Replication Server local to the `cn=monitor` branch. For detailed information, see *Summary of the Remote Replication Server Monitor Information*.
- **Replica <baseDN>**. Stores information for an instance of the replicated naming context— also known as the replica—with respect to the Directory Server and its communication with a replication server. The Replica information is what is responsible for sending and receiving changes from the replication servers. For detailed information, see *Summary of the Replica Monitor Information*.
- **Replication Server <replPort> <serverID>**. Shows the information specific to the Replication Server running, for example, on the replication port <replPort> with a server ID of <serverID>. This entry defines the replication server. For detailed information, see *Summary of the Replication Server Monitor Information*.
- **Replication Server Database <baseDN> <serverID>**. Shows information for the changelog table of replica (suffix) in the replication server. As the Replication Server receives updates from the Directory Server, it records those changes in the changelog table. For detailed information, see *Summary of the Replication Server Database Monitor Information*.
- **Replication Server Database Environment <baseDN>**. Shows the information for the database environment for the replication server backend plus the total number of records added to and deleted from the database. For detailed information, see *Summary of the Replication Server Database Environment Monitor Information*.
- **Replication Summary <baseDN>**. Shows summary information on the replication topology and the state for a particular base DN. For detailed information, see *Summary of the Replication Summary Monitor Information*.
- **Replication Changes Backend**. Shows the backend information for all replication changes. For detailed information, see *Summary of the Replication Changes Backend Monitor Information*.
- **Replication Protocol Buffer**. Shows the state of the buffer (initially 4k) for protocol operations, which is stored in thread local storage. For detailed information, see *Summary of the Replication Protocol Buffer Monitor Information*.

Replication best practices

The following are recommended best practices related to replication based on our observations in actual production environments.

About the dsreplication command-line utility

The following points involve some security practices as applies to replication. For specific questions, please contact your authorized support provider.

- **Developing Scripts**. The **dsreplication** utility maintains the history of executed **dsreplication** commands with the full command-line arguments in the `logs/tools/dsreplication.history` file. The recorded commands may be used to develop scripts to set up and configure replication.

  Scripts invoking the **dsreplication** utility in non-interactive mode should check the return code from the dsreplication process. A non-zero return code indicates some sort of failure.

  If output messages from the **dsreplication** utility are not desired, use the `--quiet` option to suppress them.

  The utility, by default, fails if one or more warnings are issued during the command execution. Warnings can be suppressed using the `--ignoreWarnings` option. For example, this option is required when using **dsreplication** with non-fully-qualified host names (for example, localhost), otherwise **dsreplication** will fail. In production environments, use of this flag is strongly discouraged.

  The **dsreplication** utility also provides an `--ignoreLock` option that specifies that the tool should continue processing in non-interactive mode or in scripts even if the replication topology has been locked by another invocation of **dsreplication**. However, this option should be used with caution.
- **Concurrent Use**. With the exception of the **dsreplication** `status` subcommand, the **dsreplication** subcommands cannot be executed concurrently. The command-line utility locks the replication topology at one or more servers to prevent accidental configuration changes caused by multiple **dsreplication** subcommands running at the same time. It is best to avoid concurrent configuration changes in general.
- **Status**. The **dsreplication** `status` subcommand requires the Replication Servers to provide monitoring information. This can lead to a delay before the output of **dsreplication** `status` is displayed. By default, **dsreplication** will display the status for all replicated domains (with the exception of the special domains of the schema and the server registry).

  It is recommended to select a particular base DN for **dsreplication** `status` if multiple base DNs are configured for replication.

  It is also recommended to avoid invoking **dsreplication** `status` too often (more than once every 15 seconds) or from multiple locations at the same time. Some of the information displayed by **dsreplication** `status` is based on monitor information that is not refreshed every time the monitor is queried.

  The `status` subcommand should not be used for collecting performance metrics. It is best to rely on replication-related information captured by the Periodic Stats Logger plugin.
- **Topology Tasks**. The **dsreplication** tool allows specifying more than one server in a topology to act as the host for other servers for the enable and initialize actions. The **--topologyFilePath** option can be used to specify a file with a series of hosts and ports in the topology. When the hosts file is used for an enable or initialize action, the servers in the file are tried sequentially until the new server is successfully enabled or added. The rest of the servers in the file are ignored. This ensures that a host server is always available. This file is generated with the **manage-topology export** command.

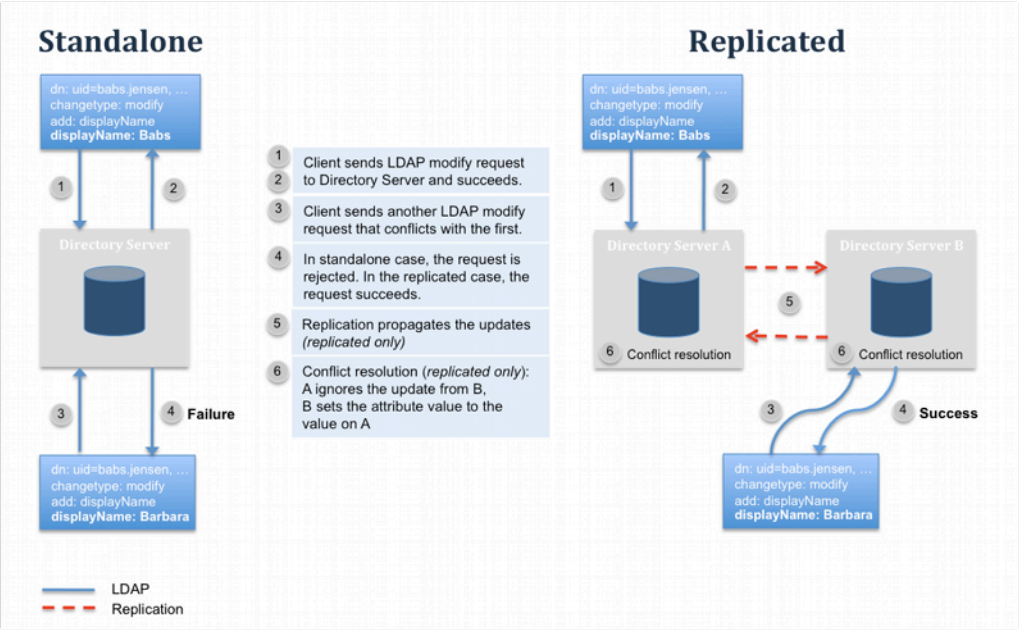
Replication conflicts

This section provides more in-depth information on replication conflicts than presented in earlier sections, so that administrators can understand the mechanisms and possible scenarios behind these conflicts.

Updates to Directory Server entries in a replication topology may happen independently, since replication guarantees only eventual consistency, not strong consistency. The eventual consistency model also means that conflicting changes can be applied at different directory server instances. In most cases, the Directory Server is able to resolve these conflicts automatically and in a consistent manner (i.e., all directory server instances in a replication topology will resolve each and every conflict the same way). However, in some scenarios, as seen below, manual administrative action is required. For any of these unresolved conflicts, the administrator is notified via administrative alerts.

On a high-level, the conflict resolution algorithm tries to resolve conflicts as if the operations causing the conflict in a distributed environment has been applied to a single directory server instance. For example, if

the same entry is added to two different directory server instances at about the same time, then once these operations have been replicated, both directory servers will keep only the entry that was added first. The following figure highlights the differences between standalone versus replicated environments.



MY TITLE Conflicting Operations in Standalone versus Replicated Environments

Types of replication conflicts

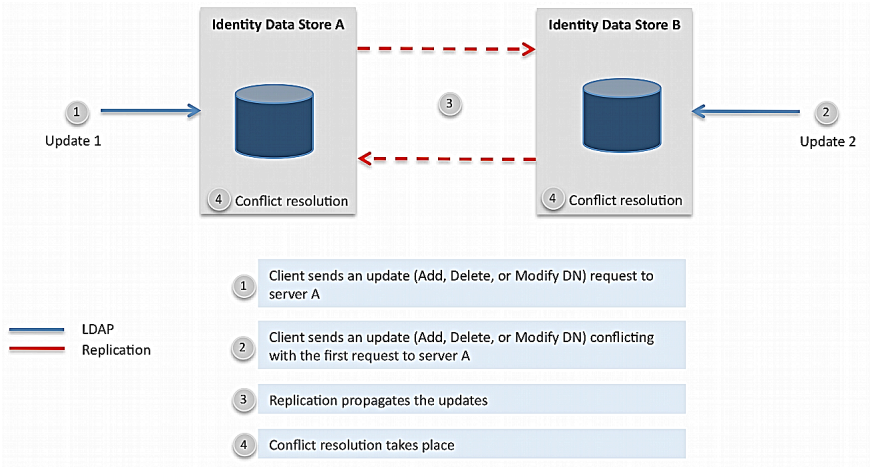
There are fundamentally two types of replication conflicts: naming and modification conflicts. Naming conflicts include operations that cause conflicts with the naming (DN) of the existing or new entries, while modification conflicts include operations that result in conflicts in the modification of attributes.

Naming conflict scenarios

For all of the naming conflict scenarios in the table below, assume the folloing:

- Update 1 was applied at Directory Server 1
- Update 2 was applied at Directory Server 2
- Update 1 occurred shortly before Update 2, so that Directory Server 2 received Update 1 after Update 2 was applied

The naming conflict scenario is illustrated in the following figure:



MY TITLE Naming Conflict Scenario

The following table shows the result of a modification conflict depending on the type of updates that occurs. The code does not compare change sequence numbers (CSNs) but applies operations in the order they were received. This may lead to inconsistent replays.

Naming Conflict Scenarios

Update 1	Update 2	Automatic Resolution?	Result of Conflict Resolution at Directory Server 2 When Update 1 is received
Modify	Delete	Yes	Modify is discarded.
Modify	Modify DN	Yes	New entry is located based on the entryUUID and Modify is applied to the renamed entry.
Delete	Delete	Yes	Delete operation is ignored.
Delete	Modify DN	Yes	Delete operation is applied to the renamed entry.
Delete of A	Add of B under A	Yes	Entry B is renamed and Entry A is deleted.
Modify DN	Delete of the same entry targeted by the Modify DN	Yes	Modify DN operation is ignored.
Modify DN with a new parent	Delete of parent	No	The entry targeted in the Modify DN operation is marked as a conflict entry.
Modify DN with a new parent	Modify DN of the parent	Yes	The entry will be moved under the new DN of the parent.
Modify DN of A with new DN B	Modify DN of C with new DN B	No	A and B will be conflict entries.
Add A	Modify DN of the parent of A	Yes	The entry is added under the new DN of the parent.
Add A	Delete of the parent of A	No	The added entry is marked as a conflict entry.

Update 1	Update 2	Automatic Resolution?	Result of Conflict Resolution at Directory Server 2 When Update 1 is received
Add A	Add A with same set of attributes	Yes	The entryUUID of the incoming Add operation applied to the existing entry.
Add A	Add A with different set of attributes (or values)	No	The existing entry is marked as a conflicting entry and the incoming Add is executed.

Modification conflict scenarios

Modification conflicts are always resolved automatically and no manual action is required. The LDAP Modify operation allows the following modification types:

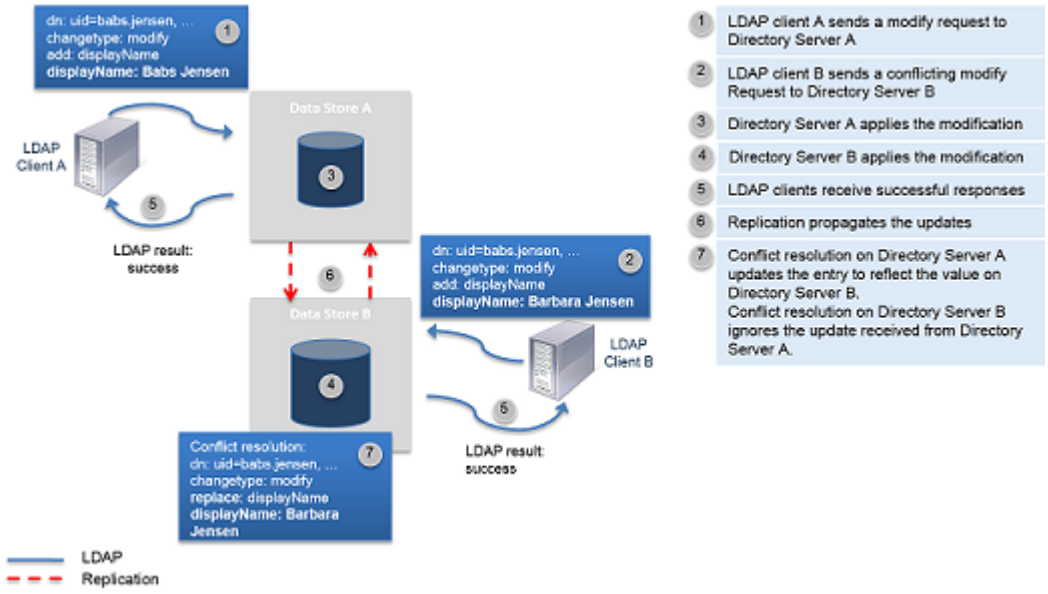
- Add of one or more values
- Delete of one or more values or the entire attribute
- Replacement of all values

Replication does not currently support the increment LDAP modification type.

For all of the operations in the table below, assume the following:

- LDAP Modify 1 was applied at Directory Server 1
- LDAP Modify 2 was applied at Directory Server 2
- LDAP Modify 1 occurred shortly before LDAP Modify 2, so that Directory Server 2 received LDAP Modify 1 after LDAP Modify 2 was applied.

The modification conflict scenario is illustrated in the figure below:



MY TITLE Modification Conflict Scenario

The following table shows the result of a modification conflict depending on the type of updates that occurs:

Modification Conflict Scenarios

Modify 1	Modify 2	Result of Conflict Resolution at Directory Server 2 When Modify 1 is received
Add of a single-value attribute	Add of the same attribute with a different value	Incoming Modify is ignored.
Delete of a single-valued attribute	Replacement of the value of the same attribute	Incoming Delete is ignored.
Replacement of a single-valued attribute	Delete of the same attribute	Incoming Replacement is ignored.
Delete some values from a multi-valued attribute	Delete some values from a multi-valued attribute	Incoming Delete is ignored.
Delete a multi-valued attribute	Delete of the same multi-valued attribute	Incoming Delete is ignored.
Delete a multi-valued attribute	Add the same multi-valued attribute	Incoming Delete is ignored.
Delete value X from a multi-valued attribute	Delete value X from the same multi-valued attribute	Incoming Delete is ignored.
Delete value X from a multi-valued attribute	Add value Y to the same multi-valued attribute	Delete of value X is applied.
Delete value X from a multi-valued attribute	Delete value Y from the same multi-valued attribute	Delete of value X is applied.
Delete value X from a multi-valued attribute	Replace all values of the same multi-valued attribute	Incoming Delete is ignored.
Add of values X and Y to a multi-valued attribute	Delete value X from the same multi-valued attribute	Only value Y is added.
Delete value X from a multi-valued attribute	Add of values X and Y to the same multi-valued attribute	Incoming Delete is ignored.

Troubleshooting replication

The following sections provide information to troubleshoot your replication deployment.

Recovering a replica with missed changes

If a server has been offline for a period of time longer than the replication purge delay, the **`dsreplication initialize`** command must be performed to bring the replica into sync with the topology.

Server startup is the only time missed changes are detected. A missed change is a change that the replica detects that it needs, but which cannot be found within any other replication server's replicationChanges backend (stored in the path server root **`/changelogDb`**). If missed changes are detected, the server enters lockdown mode, where only privileged clients can make requests. Any other server that is not missing changes can be used as a source for **`dsreplication initialize`**.

If a manual backup and restore of the server is required, then the following steps are equivalent to **`dsreplication initialize`**.

Performing a manual initialization

About this task
In the event that an online initialization is not possible, the following steps can be used to initialize a server.

Steps

1. From another server in the replication topology, back up the `userRoot`, `schema`, and `replicationChanges` backends to the `<server-root>/bak` directory. If data encryption is enabled, the `encryption-settings` backend should also be exported, as one or more encryption settings IDs may need to be imported into the new replica.

```
$ source-server-root>/bin/backup --backendID userRoot --backupDirectory \
   bak/userRoot
$ <source-server-root>/bin/backup --backendID schema --backupDirectory \
   bak/schema
$ <source-server-root>/bin/backup --backendID replicationChanges \
   --backupDirectory bak/replicationChanges
$ <source-server-root>/bin/encryption-settings export --id <id> \
   --output-file bak/exported-key
```

2. Copy the `bak` directory to the new replica.

```
$ scp -r <source-server-root>/bak/* \
    <user>@<destination-server>:<destination-server-root>/bak
```

3. Stop the server and restore the `userRoot`, `schema`, `changelog`, and `replicationChanges` backends. If the `encryption-settings` backend was exported, it should be be imported before restoring any of the backends.

```
$ <destination-server-root>bin/encryption-settings import --input-file \
   bak/exported-key --set-preferred
Enter the PIN used to encrypt the definition:
$ <destination-server-root>/bin/restore --backupDirectory bak/userRoot
$ <destination-server-root>/bin/restore --backupDirectory bak/schema
$ <destination-server-root>/bin/restore --backupDirectory \
   bak/replicationChanges
```

4. Start the server using **`bin/start-server`**.

Fixing replication conflicts

Replication conflicts can occur when an incompatible change to an entry is made on two replicas at the same time. The change is processed on one replica and then replicated to the other replica, which causes the conflict. While most conflicts are resolved automatically, some require manual action.

To fix replication conflicts, initialize the replica containing the conflicts with the data from another replica that does not have conflicts. If the database is large and the number of conflicts small, running **`ldapmodify`** against the server with the conflict will work if the command includes the Replication Repair Control specified by OID value 1.3.6.1.4.1.30221.1.5.2. The Replication Repair Control prevents the change from replicating. It also enables changing operational attribute values, which are not normally writable.

The following steps provide an example of using the Replication Repair Control to fix replication conflicts by applying change to only the server with the conflict. There are two examples: one for a modification conflict found by performing an **`ldap-diff`**, and the other for a naming conflict.

Fixing a modify conflict

Steps

1. The **bin/ldap-diff** tool can be used to isolate conflicting entries between two replicas. The following uses the tool to search across the entire base DN for any difference in user attributes, and reports the difference in **difference.ldif**. Replace the **sourceHost** value with the server that needs the adjustment.

```
$ bin/ldap-diff --sourceHost austin02.exmple.com --sourcePort 1389 \
                        --sourceBindDN "cn=Directory Manager" --
sourceBindPassword pass \
                        --targetHost austin01.example.com --targetPort 1389
 \
                        --targetBindDN "cn=Directory Manager" --
targetBindPassword
                        --baseDN "dc=example,dc=com" --outputLDIF
 difference.ldif \
                        --searchFilter "(objectclass=*)" --numPasses 3 "*"
 pass \
                        "^userPassword"
```

2. The **difference.ldif** file is in a format that can be used with **ldapmodify** to apply changes to the server that contains conflicts. The **ldap-diff** command must have been run with the **sourceHost** value as the server with conflicts. The following is an example of the contents of **difference.ldif**:

```
dn: uid=user.1,ou=people,dc=example,dc=com
                        changetype: modify
                        add: mobile
                        mobile: +1 568 232 6789
                        -
                        delete: mobile
                        mobile: +1 568 591 7372
                        -
```

3. Run **bin/ldapmodify** to correct the entries on only the server with conflicts.

```
$ bin/ldapmodify --bindPassword password -J "1.3.6.1.4.1.30221.1.5.2" \
                        --filename difference.ldif
```

Fixing a naming conflict

Steps

1. In this example, a naming conflict was encountered when the replica attempted to replay an ADD of **uid=user.200,ou=people,dc=example,dc=com**. In other words, two clients added the entry at the same time as an entry of the same name was added on another replica.

```
[18/Feb/2010:14:53:12 -0600] category=EXTENSIONS severity=SEVERE_ERROR
msgID=1880359005 msg="Administrative alert type=replication-unresolved-
conflict
id=bbd2cbaf-90a4-42af-94a8-c1a42df32fc6
class=com.unboundid.directory.server.replication.plugin.ReplicationDomain
msg='An unresolved conflict was detected for DN
 uid=user.200,ou=People,dc=example,dc=com.
The conflicting entry has been renamed to
entryuuid=69807e3d-ab27-43a3-8759-
ec0d8d6b3107+uid=user.200,ou=People,dc=example,dc=com'"
```

2. The Directory Server prepends the entryUUID to the DN of the conflicting attribute and adds a `ds-sync-conflict-entry` auxiliary object class to the entry to aid in search. For example, the following

command searches for any entry that has the `ds-sync-conflict-entry` objectclass and returns only the DNs that match the filter. You should see the conflicting entry for uid=user.200.

```
$ bin/ldapsearch --baseDN dc=example,dc=com --searchScope sub \
  "(objectclass=ds-sync-conflict-entry)" "1.1"
```

```
dn: entryuuid=69807e3d-ab27-43a3-8759-
ec0d8d6b3107+uid=user.200,ou=People,dc=example,dc=com

dn: entryuuid=523c430e-
a870-4ebe-90f8-9cd811946420+uid=user.200,ou=People,dc=example,dc=com
```

ⓘ **Note:** Conflict entries are not returned unless the `objectclass=ds-sync-conflict-entry` is present in the search filter.

3. After comparing the conflict entry with the target entry, the difference can be applied in a manner similar to the previous example using **`ldapmodify`** with the Replication Repair Control. The conflict entry can also be deleted using this command. Run **`bin/ldapmodify`** with the Replication Repair Control to make the fix. When making changes using the Replication Repair Control, the updates will not be propagated via replication. You should examine each and every replica one by one, and apply the necessary modifications using the request control.

```
$ bin/ldapmodify -J "1.3.6.1.4.1.30221.1.5.2" \
  --filename difference.ldif
```

Fixing mismatched generation IDs

A warning that multiple generation IDs were detected for a specific suffix indicates that one or more replicas need to be re-initialized. If the warning is presented from a server after an initialization, it could be that **`post-external-initialization`** was not run as part of a global change in data. Running this command will fix the situation. The **`dsreplication`** `status` command will warn when any generation IDs are different across the topology.

Replication reference

The following section shows general reference information related to replication.

Summary of the dsreplication Subcommands

A summary of the **`dsreplication`** subcommands and functions is presented in the table below.

dsreplication subcommands

Subcommand	Description
`disable`	Disables replication on the specified server for the provided base DN and removes references to this server in the other servers with which it is replicating data.
`enable`	Updates the configuration of the servers to replicate the data under the specified base DN. If one of the servers is already replicating the data under the base DN with other servers, executing this subcommand will update the configuration of all the servers (so it is sufficient to execute the command line once for each server you add to the replication topology).
`initialize`	Initializes the data under the specified base DN on the destination server with the contents on the source server.

Subcommand	Description
initialize-all	Initializes the data under the specified base DN on all servers in the replication topology with the contents on the specified server.
post-external-initialization	Used with **pre-external-initialization**, the command resets the generation ID based on the newly-loaded data. This subcommand must be called after initializing the contents of all the replicated servers using the **import-ldif** tool or **dsreplication initialize**. Specify the list of base DNs that have been initialized and provide the credentials of any of the servers that are being replicated. See the usage of the `pre-external-initialization` subcommand for more information. This subcommand only needs to be run on one of the replicas once .
pre-external-initialization	Clears the existing generation ID and removes all accumulated changes of the replicated suffix from the replication changelog database at each and every replication server. This subcommand should be used when globally restoring the replicas on all of the servers in a topology. You must specify the list of base DNs that will be initialized and provide the credentials of any of the servers that are being replicated. After calling this subcommand, initialize the contents of all the servers in the topology, then call the `post-external-initialization` subcommand. This subcommand only needs to be run on one of the replicas once .
status	Displays the status of replication domains. If no base DNs are specified as parameters, the information for all base DNs is displayed. Available options with the status subcommand are: `--showAll`, `--displayServerTable`, `--location`.

Summary of the Direct LDAP Monitor information

The following table provides a description of the attributes in the `cn=Direct LDAP Server` monitor entry. The DN for the monitor entry is as follows:

```
dn: cn=Direct LDAP Server <baseDN> <host name:ldapPort> <serverID>,cn=monitor
```

Direct LDAP Monitor Information

Monitor Attribute	Description
replica-id:<serverID>	Replica ID number.
replication-backlog	Number of changes that the replication server has not seen from the server.
missing-changes	Number of missing changes.
approximate-delay	Difference between the time of the last change that the replication server has seen from the LDAP server and the most current timestamp on the latest change on the server.
base-dn	Base DN
ssl-encryption	Flag to indicate if SSL encryption is in use.
protocol-version	Displays the replication protocol version.

Monitor Attribute	Description
generation-id	Generation ID for the base DN on the Directory Server.
restricted	Boolean that indicates whether the replication domain is restricted in an Entry Balancing Configuration with the Directory Proxy Server.
ack-sent	Number of acknowledgement messages sent to this replica (not currently used).
ack-received	Number of acknowledgement messages received from this replica (not currently used).
add-sent	Number of protocol messages with an LDAP Add sent to this replica.
add-received	Number of protocol messages with an LDAP Add received from this replica.
delete-sent	Number of protocol messages with an LDAP Delete sent to this replica.
delete-received	Number of protocol messages with an LDAP Delete received from this replica.
done-sent	Number of done messages sent to this replica. A done message indicates an end of online initialization session. If the value is non-zero, then this replica has been initialized over the replication protocol.
done-received	Number of done messages received from this replica. A done message indicates an end of online initialization session. If the value is non-zero, then this replica has completed the initialization of other replicas over the replication protocol.
entry-sent	Number of entry messages sent to this replica. Entry messages carry replicated data to initialize replicas over the replication protocol.
entry-received	Number of entry messages received from this replica. Entry messages carry replicated data to initialize replicas over the replication protocol.
error-sent	Number of error messages sent to this replica.
error-received	Number of error messages received from this replica.
heartbeat-sent	Number of heartbeat messages sent to this replica.
heartbeat-received	Number of heartbeat messages received from this replica (should always be 0, since the replica never sends a heartbeat message to the server).
initialize-request-sent	Number of initialize-request messages sent to this replica. This message is sent when another replica requested initialization of its data using the replication protocol.
initialize-request-received	Number of initialize-request messages received from this replica. This message is sent when this replica requested initialization of its data using the replication protocol from another replica.
initialize-target-sent	Number of initialize-target messages sent to this replica. This message is sent before another replica has started the initialization of this replica.

Monitor Attribute	Description
initialize-target-received	Number of initialize-target messages received from this replica. This message is sent before this replica starts the initialization of one or more replicas.
ldap-server: <hostname:ldapPort>	The hostname and port of the local replication server.
modify-sent	Number of protocol messages with an LDAP Modify sent to this replica.
modify-received	Number of protocol messages with an LDAP Modify received from this replica.
modify-dn-sent	Number of protocol messages with an LDAP Modify DN sent to this replica.
modify-dn-received	Number of protocol messages with an LDAP Modify DN received from this replica.
repl-server-start-sent	Number of replication-server-start messages sent to this replica (should never be more than 1). The Replication Server responds with this message to the start message received from the replica.
repl-server-start-received	Number of replication-server-start messages received from this replica (should always be 0).
reset-generation-id-sent	Number of reset generation ID messages received from this replica.
reset-generation-id-received	Number of reset generation ID messages sent to this replica (should always be 0).
server-start-sent	Number of server-start messages sent to this replica (should always be 0).
server-start-received	Number of server-start messages received from this replica (should never be more than 1). Server-start is the first message the replica sends after establishing a replication connection.
window-sent	Number of window messages sent to this replica. Window messages are used for cumulative acknowledgement in the replication protocol.
window-received	Number of window messages received from this replica. Window messages are used for cumulative acknowledgement in the replication protocol.
window-probe-sent	Number of window probe messages sent to this replica (should always be 0).
window-probe-received	Number of window probe messages received from this replica. The replica sends a window probe message to the server if the send window in the replica is closed and the replica is unable to publish updates to the server.
update-sent	Number of changes sent to this server.
update-received	Number of changes received from this server.
internal-connection	Indicates if the replica is in the same process as the replication server.
server-state	Displays the state of the replica. Displays the latest change number that the replica has seen from all the other replicas including itself.

Monitor Attribute	Description
consumed-update-recent-rate	Rate that the connected Directory Server is consuming updates sent by this replication server, expressed as the number of updates per second and measured over the last five seconds.
consumed-update-peak-rate	Highest rate that the connected Directory Server has consumed updates sent by this replication server, measured over a five second period since the replication server was started.
produced-update-recent-rate	Rate that the connected Directory Server has sent updates to this replication server, expressed as the number of updates per second and measured over the last five seconds.
produced-update-peak-rate	Highest rate that the connected Directory Server has sent updates to this replication server, measured over a five second period since the replication server was started.
max-send-window	Maximum number of changes that can be sent to the LDAP server before requiring an ACK.
current-send-window	Current number of changes remaining to be sent to the LDAP server before requiring an ACK.
max-rcv-window	Maximum number of changes that can be received by the LDAP server before sending an ACK.
current-rcv-window	Number of changes remaining to be received by the LDAP server before sending an ACK Server.
degraded	Indicates that the generation ID of the replica does not match the generation ID of the server. This is a temporary state, when loading data into the topology or the replica has not been initialized. Normally, it should be false.

Summary of the Indirect LDAP Server Monitor information

The following table provides a description of the attributes in the `cn=Indirect LDAP Server` monitor entry. These attributes provide information about a Directory Server that is connected to a different replication server in the topology.

```
dn: cn=Indirect LDAP Server <baseDN> <serverID>,cn=monitor
```

Indirect LDAP Server Monitor Information

Monitor Attribute	Description
replica-id: <serverID>	ID number identifying the replica.
base-dn: <baseDN>	Base DN
connected-to: Remote Repl Server <baseDN> <host name:replPort> <replID>	Replication server to which the directory server is connected.
replication-backlog	Number of changes that the replication server has not seen from the server.
approximate-delay	Amount of time between the last change seen by this Directory Server and the most recent change seen by the remote replication server. This value is the amount of time between the time stamps, not the amount of time required to synchronize the two servers.

Monitor Attribute	Description
generation-id	Generation ID for this suffix on this remote replication server.
consumed-update-recent-rate	Rate that the connected Replication Server is consuming updates sent by this replication server, expressed as the number of updates per second and measured over the last five seconds.
consumed-update-peak-rate	Highest rate that the connected Replication Server has consumed updates sent by this replication server, measured over a five second period since the replication server was started.

Summary of the Remote Replication Server Monitor information

The following table provides a description of the attributes in the `cn=Remote Repl Server` monitor entry. The DN for the monitor entry is as follows:

```
dn: cn=Remote Repl Server <baseDN> <host name:replPort> <serverID>,cn=monitor
```

Remote Replication Server Monitor Information

Monitor Attribute	Description
replication-server: <host name>:<repl port>	Host name and replication port number of the Replication Server.
replication-server-id:<serverID>	Server ID for the Replication Server.
available	Indicates if the remote replication server is available or not. Values: true or false.
sending-paused	Indicates if sending is paused. Values: true or false.
receiving-paused	Indicates if receiving is paused. Values: true or false.
wan-gateway-priority	Specifies the WAN Gateway priority of the remote replication server.
is-wan-gateway	Indicates if the remote replication server is a WAN Gateway. Values: true or false.
wan-gateway-desired	Indicates if the remote replication server is a desired gateway. Values: true or false. This entry together with the `is-wan-gateway` property indicates the desired state. ▪ **is-wan-gateway=false, wan-gateway-desire=false**: Indicates another server with a higher gateway priority exists or the gateway priority is set to disabled. ▪ **is-wan-gateway=false, wan-gateway-desire=true**: Indicates that the remote replication server wants to be a WAN gateway. ▪ **is-wan-gateway=true, wan-gateway-desire=false**: Indicates that the remote replication server wants to give up its role as a WAN gateway. ▪ **is-wan-gateway=false, wan-gateway-desire=true**: Indicates the remote replication server wants to remain as a gateway server.
base-dn	base DN
ssl-encryption	Flag to indicate if SSL encryption is in use.
protocol-version	Replication protocol version.
generation-id	Generation ID for this suffix on this remote replication server.

Monitor Attribute	Description
restricted	Indicates that remote replication server is in an entry-balancing deployment.
add-sent	Number of protocol messages with an LDAP Add sent to the remote replication server.
add-received	Number of protocol messages with an LDAP Add received from the remote replication server.
delete-sent	Number of protocol messages with an LDAP Delete sent to the remote replication server.
delete-received	Number of protocol messages with an LDAP Delete received from the remote replication server.
done-sent	Number of done messages sent to the remote replication server. A done message indicates an end of online initialization session.
done-received	Number of done messages received from the remote replication server. A done message indicates an end of online initialization session.
entry-sent	Number of entry messages sent to the remote replication server. Entry messages carry replicated data to initialize replicas over the replication protocol.
entry-received	Number of entry messages received from the remote replication server. Entry messages carry replicated data to initialize replicas over the replication protocol.
error-sent	Number of error messages sent to the remote replication server.
error-received	Number of error messages received from the remote replication server.
heartbeat-sent	Number of heartbeat messages sent to the remote replication server.
heartbeat-received	Number of heartbeat messages received from the remote replication server.
initialize-request-sent	Number of initialize-request messages sent to the remote replication replication server. This message is used during online initialization from one replica to another.
initialize-request-received	Number of initialize-request messages received from the remote replication server. This message is used during online initialization from one replica to another.
initialize-target-sent	Number of initialize-target messages sent to the remote replication server. This message is used before online initialization from one replica to another.
initialize-target-received	Number of initialize-target messages received from the remote replication server. This message is used before online initialization from one replica to another.
modify-sent	Number of protocol messages with an LDAP Modify sent to the remote replication server.
modify-received	Number of protocol messages with an LDAP Modify received from the remote replication server.

Monitor Attribute	Description
modify-dn-sent	Number of protocol messages with an LDAP Modify DN sent to the remote replication server.
modify-dn-received	Number of protocol messages with an LDAP Modify DN received from the remote replication server.
monitor-sent	Number of monitor messages sent to the remote replication server. This message is primarily used when communicating with directory servers running a prior release.
monitor-received	Number of monitor messages received from the remote replication server. This message is primarily used when communicating with directory servers running a prior release.
monitor-request-sent	Number of monitor requests sent to the remote replication server. The receiving server will respond with a monitor message that includes server's information about the state of the topology.
monitor-request-received	Number of monitor requests received from the remote replication server. This server will respond with a monitor message that includes server's information about the state of the topology.
monitor-v2-sent	Number of monitor messages sent to the remote server. This monitor message is only used with Directory Server v3.5 or later.
monitor-v2-received	Number of monitor messages received from the remote server. This monitor message is only used with Directory Server v3.5 or later.
pause-sending-updates-sent	Number of pause-sending-updates messages sent to the remote server. The remote server must stop sending update messages to this server when receiving this message.
pause-sending-updates-received	Number of pause-sending-updates messages received from the remote server. This server will stop sending update messages to the remote server upon receiving this message.
repl-server-start-sent	Number of replication-server-start messages sent to the remote replication server (should never be more than 1). The Replication Server responds with this message to the replication-server-start message received from remote replication servers.
repl-server-start-received	Number of replication-server-start messages received from the remote replication server (should never be more than 1) window-sent: the number of window messages sent to the remote replication server. Window messages are used for cumulative acknowledgement in the replication protocol.
reset-generation-id-sent	Number of reset generation ID messages received from the remote replication server. This message is sent before and after the data is initialized in the topology.
reset-generation-id-received	Number of reset generation ID messages sent to the remote replication server. This message is sent before and after the data is initialized in the topology.
server-info-sent	Number of replication server information messages sent to the remote replication server. This message tells other replication servers about the replicas directly connected to the sending server. This message is also used to distribute information about the location of replicas.

Monitor Attribute	Description
server-info-received	Number of replication server information messages received from the remote replication server. This message tells other replication servers about the replicas directly connected to the sending server. This message is also used to distribute information about the location of replicas.
set-source-location-sent	Number of set-source-locations messages sent to the remote replication server. This message is used by WAN gateway servers to request update messages from additional locations.
set-source-location-received	Number of set-source-locations messages sent to the remote replication server. This message is used by WAN gateway servers to request update messages from additional locations.
start-sending-updates-sent	Number of start-sending-updates messages sent to the remote replication server. The remote server may only start sending updates to this server after receiving this message.
start-sending-updates-received	Number of start-sending-updates messages received from the remote server. Sending update messages to the remote server may only start after receiving this message.
window-sent	Number of window messages sent to the remote replication server. Window messages are used for cumulative acknowledgement in the replication protocol.
window-received	Number of window messages received from the remote replication server. Window messages are used for cumulative acknowledgement in the replication protocol.
messages-sent	Total number of messages sent.
messages-received	Total number of messages received.
update-sent	Total number of updates sent.
update-received	Total number of updates received.
server-state	Displays the server state of the remote replication server. It displays the last change seen on the remote replication server.
consumed-update-recent-rate	Rate that the connected Directory Server is consuming updates sent by this replication server, expressed as the number of updates per second and measured over the last five seconds.
consumed-update-peak-rate	Highest rate that the connected Directory Server has consumed updates sent by this replication server, measured over a five second period since the replication server was started.
produced-update-recent-rate	Rate that the connected Directory Server has sent updates to this replication server, expressed as the number of updates per second and measured over the last five seconds.
produced-update-peak-rate	Highest rate that the connected Directory Server has sent updates to this replication server, measured over a five second period since the replication server was started.
max-send-window	Maximum number of changes that can be sent to the remote replication server before requiring an ACK.
current-send-window	Number of changes remaining to be sent to the remote replication server before requiring an ACK.

Monitor Attribute	Description
max-rcv-window	Maximum number of changes that can be received from the remote replication server before sending an ACK.
current-rcv-window	Number of changes remaining to be received from the remote replication server before sending an ACK.
degraded	Indicates that the generation ID of the replica does not match the generation ID of the remote replication server. This is a temporary state, when loading data into the topology or the replica has not been initialized. Normally, it should be false.

Summary of the Replica Monitor information

The following table provides a description of the attributes in the `cn=Replica` monitor entry for a specific base DN.

```
dn: cn=Replica <baseDN>,cn=monitor
```

Indirect LDAP Server Monitor Information

	Description
	Specified base DN. The monitor entries track your company's base DN (or base-DN: <baseDN>`dc=example,dc=com`), `cn=schema`base-DN: , and `cn=topology,cn=config`.
connected-to: Replication Server <replPort> <serverID>	Replication port number and server ID of the replication server to which this LDAP Server is connected. The first number is the replication server port number and the second number is the `server-id` of the replication server.
lost-connections	Number of times the Directory Server has lost connection to a replication server.
received-updates	Number of updates that the Directory Server Replica has received from the connected replication server.
sent-updates	Number of updates sent to the replication server.
pending-updates	Number of updates pending to send to the replication server.
replayed-updates	Total number of updates from the replication server that have been replayed for this replica.
replayed-updates-ok	Number of updates for this replica that have been successfully replayed with no conflicts.
replayed-update-failed	Number of updates for this replica that were successfully replayed after automatically resolving a modify conflict.
resolved-modify-conflicts	Number of updates for this replica that were successfully resolved after a modify conflict.
resolved-naming-conflicts	Number of updates for this replica that were successfully resolved after a naming conflict.
unresolved-naming-conflicts	Number of updates for this replica that could not be replayed due to an unresolvable naming conflict.
replica-id	Server ID for this replica.

	Description
max-rcv-window	Maximum number of changes that the Directory Server Replica can receive at a time before sending an acknowledgment back to the replication server.
current-rcv-window	Current received window size for this replica.
max-send-window	Maximum number of changes that the Directory Server Replica can send at a time to the replication server before requiring an ACK.
current-rcv-window	Number of changes remaining to be received from the replication server before it must send an ACK.
max-send-window	Maximum number of changes that the Directory Server Replica can send at a time to the replication server before requiring an ACK.
current-send-window	Number of changes remaining to be sent to the replication server before requiring an ACK.
ssl-encryption	Flag to indicate if SSL encryption is in use.
generation-id	Generation ID for this suffix on the Directory Server.
replication-backlog	Number of changes that are from this replica.

Summary of the Replication Server Monitor information

The following table provides a description of the attributes in the `cn=Replication Server` monitor entry.

```
dn: cn=Replication Server <baseDN> <replServerID>,cn=monitor
```

Replication Server Monitor Information

Monitor Attribute	Description
replication-server-id	Server ID for the Replication server ID.
replication-server-port	Port number on which the replication server listens for communication from other servers.
base-dn: <baseDN>	Indicates the suffix to which this replication server database applies.
Generation IDs by Base DN	List of generation IDs for each base DN on the server.
num-outgoing-replication-server-connections	Number of outgoing connections from the replication server.
num-incoming-replication-server-connections	Number of incoming connections into the replication server.
num-incoming-replica-connections	Number of incoming connections to the replica.

Summary of the Replication Server Database Monitor information

The following table provides a description of the attributes in the `cn=Replication Server database` monitor entry.

```
dn: cn=Replication Server database <baseDN> <replServerID>,cn=monitor
```

Replication Server Database Monitor Information

Monitor Attribute	Description
database-replica-id: <replicaID>	Specifies the replication server ID.
base-dn: <baseDN>	Indicates the suffix to which this replication server database applies.
first-change	First change number that is in this replication database table for this suffix from this `server-id`. For example, an example entry looks like the following: `0000012209EA622C390D00000002 Mon Jun 22 16:41:11 CDT 2011`
last-change	Last change number that is in this replication database table for this suffix from this `server-id`. For example, an example entry looks like the following: `0000012209EA622C390D00000002 Mon Jun 22 16:41:11 CDT 2011`
queue-size	Number of changes in the replication server queue waiting to be sent to this remote replication server database.
queue-size-bytes	Size in bytes of all the messages waiting in the queue.
records-added	Displays the number of records changed or added to the DIT.
records-removed	Displays the number of records removed from the DIT.

Summary of the Replication Server Database Environment Monitor information

The following table provides a description of the attributes in the `cn=Replication Server Database Environment` monitor entry, which includes the environment variables associated with the Oracle Berkeley Database Java Edition backend.

```
dn: cn=Replication Server Database Environment,cn=monitor
```

Replication Server Database Environment Monitor Information

Monitor Attribute	Description
je-version	Current version of the Oracle Berkeley Java Edition.
current-db-cache-size	Current DB cache size.
max-db-cache-size	Maximum DB cache size.
db-cache-percent-full	Percentage of the cache used by the Directory Server.
db-directory	Directory that holds the changelogDb file.
db-on-disk-size	Size of the DB on disk.
cleaner-backlog	Number of log files that must be cleaned for the cleaner to meet its target utilization.
random-read-count	Number of disk reads which required repositioning the disk head more than 1MB from the previous file position.
random-write-count	Number of disk writes which required repositioning the disk head by more than 1MB from the previous file position.

Monitor Attribute	Description
sequential-read-count	Number of disk reads which did not require repositioning the disk head more than 1MB from the previous file position.
sequential-write-count	Number of disk writes which did not require repositioning the disk head by more than 1MB from the previous file position.
nodes-evicted	Accumulated number of nodes evicted.
active-transaction-count	Number of currently active transactions.
num-checkpoints	Number of checkpoints. A checkpoint is a process that writes to your log files all the internal BTree nodes and structures modified as a part of write operations to your log files to facilitate a quick recovery.
checkpoint-in-progress: false	Indicates if a checkpoint is in progress.
total-checkpoint-duration-millis	Total time in milliseconds for all checkpoints.
average-checkpoint-duration-millis	Average time in milliseconds for all checkpoints.
last-checkpoint-duration-millis	Duration in milliseconds of the last checkpoint run.
last-checkpoint-start-time	Start time of the last checkpoint.
last-checkpoint-stop-time	Stop time of the last checkpoint.
millis-since-last-checkpoint	Time in milliseconds since the last checkpoint.
read-locks-held	Total read locks currently held.
write-locks-held	Total write locks currently held.
transactions-waiting-on-locks	Total transactions waiting for locks
je-env-stat-AdminBytes	Number of bytes of JE cache used for log cleaning metadata and other administrative structure.
je-env-stat-BufferBytes	Total memory currently consumed by log buffers, in bytes.
je-env-stat-CacheDataBytes	Total memory of cache used for data.
je-env-stat-CacheTotalBytes	Total amount of JE cache in use, in bytes.
je-env-stat-CleanerBacklog	Number of files to be cleaned to reach the target utilization.
je-env-stat-CursorsBins	Number of bottom internal nodes (BINs) encountered by the INCompressor that had cursors referring to them when the compressor ran. The compressor thread cleans up the internal BTree as records are deleted to ensure unused nodes are not present.
je-env-stat-DataBytes	Amount of JE cache used for holding data, keys and internal Btree nodes, in bytes.
je-env-stat-DbClosedBins	Number of bins encountered by the INCompressor that had their database closed between the time they were put on the compressor queue and when the compressor ran.
je-env-stat-EndOfLog	Location of the next entry to be written to the log.
je-env-stat-InCompQueueSize	Number of entries in the INCompressor queue when the getStats() call was made.
je-env-stat-LastCheckpointEnd	Location in the log of the last checkpoint end.
je-env-stat-LastCheckpointId	ID of the last checkpoint.

Monitor Attribute	Description
je-env-stat-lastCheckpointStart	Location in the log of the last checkpoint start.
je-env-stat-lockBytes	Number of bytes of JE cache used for holding locks and transactions.
je-env-stat-NBINsStripped	Number of BINS stripped by the evictor.
je-env-stat-NCacheMiss	Total number of requests for database objects which were not in memory.
je-env-stat-NCheckpoints	Total number of checkpoints run so far.
je-env-stat-NCleanerDeletions	Number of cleaner file deletions this session.
je-env-stat-NCleanerEntriesRead	Accumulated number of log entries read by the cleaner.
je-env-stat-NCleanerRuns	Number of cleaner runs this session.
je-env-stat-NClusterLNsProcessed	Accumulated number of leaf nodes (LNs) processed because they qualify for clustering.
je-env-stat-NDeltaINFlush	Accumulated number of delta internal nodes (INs) flushed to the log.
je-env-stat-NEvictPasses	Number of passes made to the evictor.
je-env-stat-NFSyncRequests	Number of fsyncs requested through the group commit manager. **`Fsync()`** synchronizes the file system after a write or a transaction.
je-env-stat-NFSyncTimeouts	Number of fsync requests submitted to the group commit manager which timed out.
je-env-stat-NFSyncs	Number of fsyncs issued through the group commit manager.
je-env-stat-NFileOpens	Number of times a log file has been opened.
je-env-stat-NFullBINFlush	Accumulated number of full bottom internal nodes (BINS) flushed to the log.
je-env-stat-NFullINFlush	Accumulated number of full INs flushed to the log.
je-env-stat-NINsCleaned	Accumulated number of INs cleaned.
je-env-stat-NINsDead	Accumulated number of INs that were not found in the tree anymore (deleted).
je-env-stat-NINsMigrated	Accumulated number of INs migrated.
je-env-stat-NINsObsolete	Accumulated number of INs obsolete.
je-env-stat-NLNQueueHits	Accumulated number of LNs processed without a tree lookup.
je-env-stat-NLNsCleaned	Accumulated number of LNs cleaned.
je-env-stat-NLNsDead	Accumulated number of LNs that were not found in the tree anymore (deleted).
je-env-stat-NLNsLocked	Accumulated number of LNs encountered that were locked.
je-env-stat-NLNsMarked	Accumulated number of LNs that were marked for migration during cleaning.
je-env-stat-NLNsMigrated	Accumulated number of LNs migrated.
je-env-stat-NLNsObsolete	Accumulated number of LNs obsolete.
je-env-stat-NLogBuffers	Number of log buffers currently instantiated.
je-env-stat-NMarkedLNsProcessed	Accumulated number of LNs processed because they were previously marked for migration.

Monitor Attribute	Description
je-env-stat-NNodesExplicitlyEvicted	Accumulated number of nodes evicted.
je-env-stat-NNodesScanned	Accumulated number of nodes scanned to select the eviction set.
je-env-stat-NNodesSelected	Accumulated number of nodes selected to evict.
je-env-stat-NNotResident	Number of requests for database objects not contained within the in memory data structures.
je-env-stat-NOpenFiles	Number of files currently open in the file cache.
je-env-stat-NPendingLNsLocked	Accumulated number of pending LNs that could not be locked for migration because of a long duration application lock.
je-env-stat-NPendingLNsProcessed	Accumulated number of LNs processed because they were previously locked.
je-env-stat-NRandomReadBytes	Number of bytes read which required repositioning the disk head more than 1MB from the previous file position.
je-env-stat-NRandomReads	Number of disk reads which required repositioning the disk head more than 1MB from the previous file position.
je-env-stat-NRandomWriteBytes	Number of bytes written which required repositioning the disk head more than 1MB from the previous file position.
je-env-stat-NRandomWrites	Number of disk writes which required repositioning the disk head by more than 1MB from the previous file position.
je-env-stat-NRepeatFaultReads	Number of reads which had to be repeated when faulting in an object from disk because the read chunk size controlled by `je.log.faultReadSize` is too small.
je-env-stat-NRepeatIteratorReads	Number of times we try to read a log entry larger than the read buffer size and can't grow the log buffer to accommodate the large object.
je-env-stat-NRootNodesEvicted	Accumulated number of database root nodes evicted.
je-env-stat-NSequentialReadBytes	Number of bytes read which did not require repositioning the disk head more than 1MB from the previous file position.
je-env-stat-NSequentialReads	Number of disk reads which did not require repositioning the disk head more than 1MB from the previous file position.
je-env-stat-NSequentialWriteBytes	Number of bytes written which did not require repositioning the disk head more than 1MB from the previous file position.
je-env-stat-NSequentialWrites	Number of disk writes which did not require repositioning the disk head by more than 1MB from the previous file position.
je-env-stat-NSharedCacheEnvironments	Number of environments using the shared cache.
je-env-stat-NTempBufferWrites	Number of writes which had to be completed using the temporary marshalling buffer because the fixed size log buffers specified by `je.log.totalBufferBytes` and `je.log.numBuffers` were not large enough.
je-env-stat-NToBe-CleanedLNsProcessed	Accumulated number of LNs processed because they are soon to be cleaned.
je-env-stat-NonEmptyBins	Number of non-empty bins.

Monitor Attribute	Description
je-env-stat-ProcessedBins	Number of bins that were successfully processed by the INCompressor.
je-env-stat-RequiredEvictBytes	Number of bytes that must be evicted to get within the memory budget.
je-env-stat-SharedCacheTotalBytes	Total amount of the shared JE cache in use, in bytes.
je-env-stat-SplitBins	Number of bins encountered by the INCompressor that were split between the time they were put on the compressor queue and when the compressor ran.
je-env-stat-TotalLogSize	Approximation of the current total log size in bytes.
je-env-stat-NOwners	Total lock owners in the lock table.
je-env-stat-NReadLocks	Total read locks currently held.
je-env-stat-NRequests	Total number of lock requests to date.
je-env-stat-NTotalLocks	Total locks currently in the lock table.
je-env-stat-NWaiters	Total transactions waiting for locks.
je-env-stat-NWaits	Total number of lock waits to date.
je-env-stat-NWriteLocks	Total write locks currently held.
je-env-stat-LastCheckpointTime	Time of the last checkpoint.
je-env-stat-LastTxnId	Last transaction ID allocated.
je-env-stat-NAborts	Number of transactions that have aborted.
je-env-stat-NActive	Number of transactions that are currently active.
je-env-stat-NBegins	Number of transactions that have begun.
je-env-stat-NCommits	Number of transactions that have committed.
je-env-stat-NXAAborts	Number of XA transactions that have aborted.
je-env-stat-NXACommits	Number of XA transactions that have committed.
je-env-stat-NXAPrepares	Number of XA transactions that have been prepared.

Summary of the Replication Summary Monitor information

The following table provides a description of the attributes in the `cn=Replication Summary` monitor entry.

```
dn: cn=Replication Summary <baseDN>,cn=monitor
```

Replication Summary Monitor Information

Monitor Attribute	Description
base-dn:<baseDN>	Base DN summary.
replica: replica-id, ldap-server connected-to, generation-id, replication-backlog, recent-update-rate, peak-update-rate, age-of-oldest- backlog-change	Summary information for each replica in the topology. This entry appears for each replica in the topology with its own respective properties.

Monitor Attribute	Description
replication-server: server-id, server, generation-id, status, last-connected, last-failed, failed-attempts, attributes.	Summary information for each remote replication server only in the topology. This entry appears for each Replication Server in the topology with its own respective `serverID` and `server` properties.
update-queue: id, max-count, current-count, max-size, current-size, polling-source, polling-source-changed	Summary information for each update queue on a server: ▪ **id**. The ID of the receiving replica or replication server ▪ **max-count**. The maximum number of update messages that the sending replication server will keep in memory for the receiving replica or replication server. If the receiver cannot accept messages fast enough for any reason (high load, network latency, etc), then this queue will fill up. When that happens, the sending replication server will read update messages from the changelog backend. This slows down the update processing considerably. ▪ **current-count**. The number of update messages currently on the queue that have not been sent to the receiving replica or replication server. Every time the sending replication server sends an update to the receiving replica or replication server, this counter is decremented. ▪ **max-size**. The maximum total size (in bytes) of update messages that may be in the queue. This queue is capped by both the maximum count (`max-count`) and the `max-size` setting, whichever is reached first. ▪ **current-size**. The total size of update messages currently on the queue that have not been sent to the receiving replica or replication server. Every time the sending replication server sends an update to the receiving replica or replication server, this value is decremented by the size of the published update message. ▪ **polling-source**. Either 'memory' or 'db'. If set to 'memory', the sending replication server relies only on the in-memory queue to push update messages to the receiving replica or replication server. If set to 'db', update messages are read and sorted from the changelog database, which is significantly slower than publishing updates from the in-memory queue. ▪ **polling-source-changed**. The total number of times the polling-source attribute has changed value (either from 'db' to 'memory' or from 'memory' to 'db'). If this value changes very frequently, then the queue size setting is probably too low.

Summary of the replicationChanges Backend Monitor information

The following table provides a description of the attributes in the `cn=replicationChanges Backend` monitor entry.

```
dn: cn=replicationChanges Backend,cn=monitor
```

replicationChanges Backend Monitor Information

Monitor Attribute	Description
ds-backend-id	ID descriptor for the backend. Typically, this will be "replicationChanges".

Monitor Attribute	Description
ds-backend-base-dn	Base DN for the backend. Typically, this will be `cn=replicationChanges`.
ds-backend-is-private	Flag to indicate if the backend is private.
ds-backend-entry-count	Entry count for the backend.
ds-base-dn-entry-count	Entry count for the base DN and the specified base DN.
ds-backend-writability-mode	Flag to indicate if the backend is writable or not.

Summary of the Replication Protocol Buffer Monitor information

The following table provides a description of the attributes in the `cn=Replication Protocol` Buffer monitor entry. The monitors provide information on the state of the buffer for protocol operations, which is kept in the local storage.

```
dn: cn=Replication Protocol Buffer,cn=monitor
```

Replication Protocol Buffer Monitor Information

Monitor Attribute	Description
saved-buffers	Number of protocol buffers. Initial buffer size is 4k.
reallocations	Number of times the buffers had to be reallocated due to insufficient size.
large-buffer-creates	Number of times a buffer larger than 512k was requested.
large-buffer-evictions	Number of times a buffer was removed from the thread local storage, because it has grown above 512k.

Advanced topics reference

This chapter presents background reference information covering advanced replication topics.

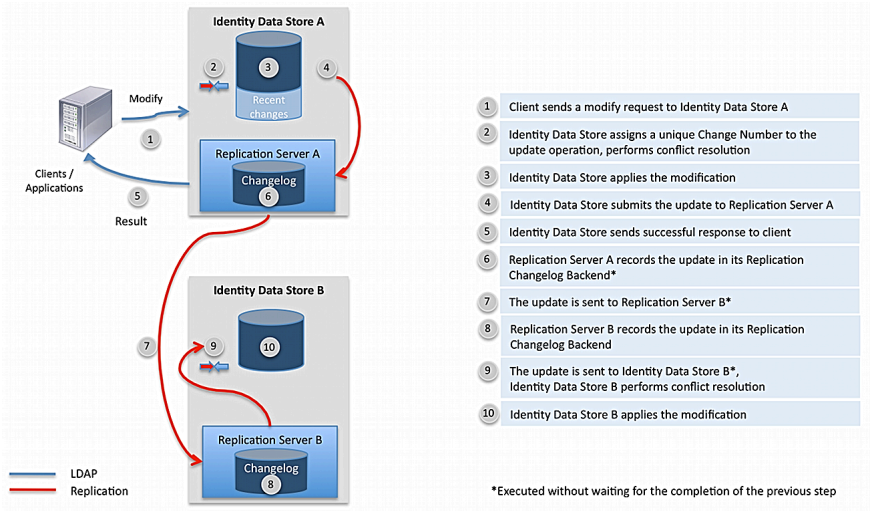
About the replication protocol

Replication communicates using a proprietary binary protocol that is implemented on top of the TCP/IP protocol using SSL encryption. Some protocol messages are used for administrative purposes (such as WAN Gateway server negotiation or flow control), some carry updates to replicated data, while others are directed to all servers for monitoring requests.

In a replicated topology, each participating Directory Server is connected to every other server via the replication server port in order to monitor health. Servers which share the same location setting are also connected to rapidly replicate changes and lastly the WAN Gateway servers are all interconnected to replicate changes across locations.

Directory Servers keep connections open as long as possible to reduce the communication latency when messages are exchanged. Heartbeat messages are transmitted on a regular basis to detect a network failure or an unresponsive directory server as early as possible. Heartbeat messages also prevent idle connections from being closed by firewalls.

The following detailed communication flow will be used to describe major components of replication. This illustration is the expanded view of figure shown in the Overview section.



MY TITLE Replication Communication Flow

Step 1. Client sends a Modify request to Directory Server A.

Step 2. Directory Server A assigns a unique change number to the operation. Conflict resolution is executed to see if the Modify request is in conflict with the existing attribute types or values in the existing entry. The change number is assigned before the Directory Server backend is updated so that the arrival order of client requests can be preserved. Historical data in the target entry is updated to reflect the change. Note that historical data is only updated for ADD and MODIFY operations.

Step 3. Directory Server applies the modifications in the corresponding backend.

Step 4. If the MODIFY operation successfully completes, then the Directory Server will submit the update to its embedded *Replication Server*. The Replication Server is a component within the Directory Server process responsible for propagating updates to and from the replication topology. The Directory Server itself only communicates with a single replication server, whereas the replication server component is connected to all other replication servers. If the Directory Server process exits unexpectedly and some updates have not been passed to the Replication Server, the backend has the ability to recover the last 50,000 recent changes that were processed at this server, guaranteeing that these changes can be replicated when the server starts up. The figure above also shows that replication protocol is used not just between replication servers but also between the Directory Server and the Replication Server.

Step 5. The response is sent to the client. In this example, a successful response is assumed.

Step 6. The Replication Server records the update in its own Changelog backend (i.e., backend ID of `replicationChanges`) and on disk with the path to `changelogDb` under the server root. The Replication Changelog backend keeps track of updates at each and every Directory Server in the replication topology. When a Directory Server joins the replication topology after being disconnected for some reason, updates from the Replication Changelog backend are re-sent to this Directory Server. Old records from the Replication Changelog backend are purged, which by default removes records older than 24 hours. If the backend does not contain all of the records that another Directory Server needed when rejoining the replication topology, then the replicated data set in the Directory Server must be re-initialized. In this case, the Directory Server enters lockdown mode and an administrative alert is sent.

Step 7. The Replication Server submits the update to the replication server component in Directory Server B. If there were more Directory Servers in this example, the Replication Server would submit the update to all the other replication servers in the same location.

Step 8. Just like in Step 6, the Replication Server component receiving an update inserts the change into its Replication Changelog backend.

Step 9. The update is forwarded to the Replica in Directory Server B. Conflict resolution is executed to see if the Modify request is in conflict with the existing attribute types or values in the existing entry.

Step 10. The Directory Server applies the modification in the corresponding backend. The Recent Changes database is not updated, because only updates that originated at this Directory Server are recorded in the Recent Changes database.

Change number

As seen in the previous Figure, the Directory Server assigns a unique change number to each update operation (specifically, ADD, DELETE, MODIFY, or MODIFY DN operations) to track each request when received from a client. The change number not only identifies each and every update, but it also allows ordering updates to the replicated data the same way on each Directory Server. The change number is composed of the following multiple fields:

- **Timestamp** that identifies when the update was made. The timestamp is time-zone-independent and recorded with millisecond resolution.
- **Server ID** that uniquely identifies the Directory Server where the update was made.
- **Sequence number** that defines the order of the updates received from external clients at a particular directory server.

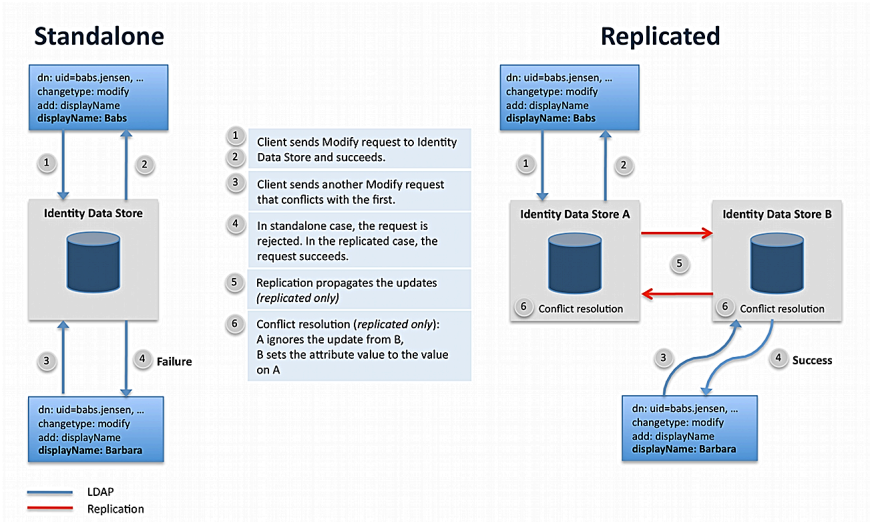
The replication protocol also sets a virtual clock that eliminates the need for synchronized time on servers. For troubleshooting purposes, however, it is still recommended to keep the system clocks synchronized.

Conflict resolution

The eventual-consistency model employed in replication introduces a window where conflicting updates targeting the same entry may be applied at two different Directory Servers. In general, two updates to the same Directory Server are in conflict if the update that arrived later fails. Conflict resolution, when possible, corrects conflicts introduced by clients automatically. There are some exceptions, however, when manual administrative action is required. For example, adding an entry in one replica and deleting the parent of this entry on another replica simultaneously will introduce a conflict that requires manual action. In a carefully implemented deployment, the risk of introducing conflicts that require manual action can be significantly reduced or even eliminated.

The conflict resolution algorithm in the PingDirectory Server uses a mechanism that orders all updates in the replication topology. Each update in the Directory Server is assigned a unique change number. The change number is attached to each update propagated via replication and allows each Directory Server to order updates exactly the same way.

Consider the following example that results in a conflict: add a single-valued attribute with different values to an entry concurrently at two Directory Servers (shown in the figure below). It is easy to see that the second operation would fail if a client attempted to add the same attribute to the same entry at the same Directory Server. In a replicated environment, the conflict is not immediately seen if these updates are applied concurrently at two different Directory Servers. The conflict is handled only after replication propagates the updates. The Directory Servers resolve the conflict independently of the other server. On one Directory Server, the entry will be updated to reflect the correct value; on the other Directory Server, the value will stay the same. As result, each Directory Server will independently resolve the conflict the same way based on the ordering of the updates. This example is illustrated below:



MY TITLE Conflict Resolution Process Flow

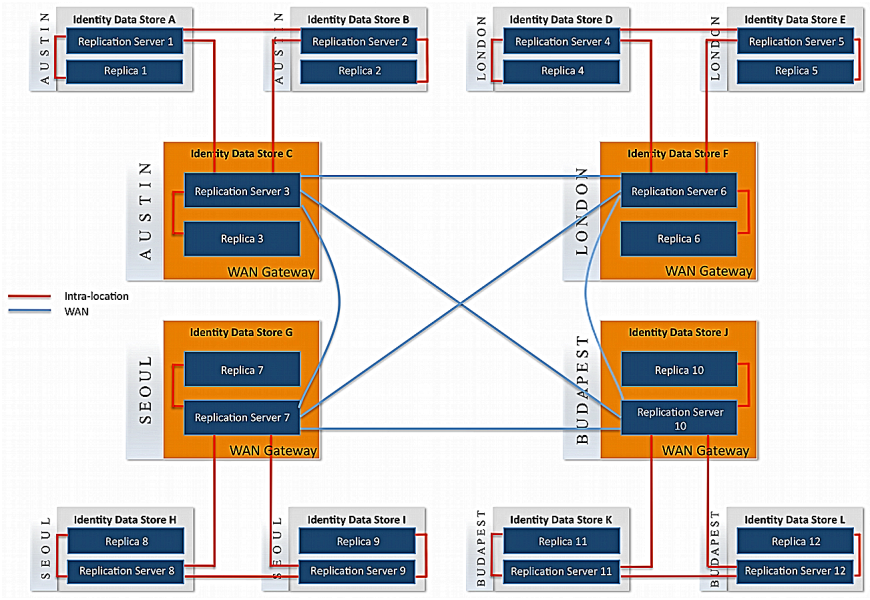
WAN-friendly replication

Many multi-national corporations that have data centers in different countries must minimize latency over WAN to ensure acceptable performance for their client applications. To minimize WAN latency, the Directory Server assigns one of two roles to the replication servers: the role of standard replication transmitting updates to the other co-located replication servers; the other role, a WAN-dedicated replication server designed to send updates to other WAN-designated replication servers in other locations. This two-role system minimizes WAN traffic by pushing all replication updates onto the connected replication servers that are designated as WAN Gateway Servers. Only the designated WAN Gateway Servers can transmit the update messages to other connected WAN Gateway servers at other locations.

WAN Gateway Server

The Directory Server's replication mechanism relies on the server's location information to reduce protocol traffic on WAN links. During protocol negotiation, the replication server with the highest WAN Gateway priority (priority 1 indicates the highest priority) automatically assumes the role as the WAN Gateway Server for that particular location. The Gateway Server's main function is to route update messages from other non-gateway servers at the same location to remote WAN Gateway servers at other locations. Similarly, at the destination point, the replication server with the WAN Gateway role will receive update messages from other WAN gateway servers at other locations and push them out to all replication servers at the current location. This setup ensures that all WAN communication flows through the WAN Gateway Servers.

The figure below shows a basic connection configuration for updates. Keep in mind that all of the replication servers are fully connected to each other for monitoring or server negotiation purposes.



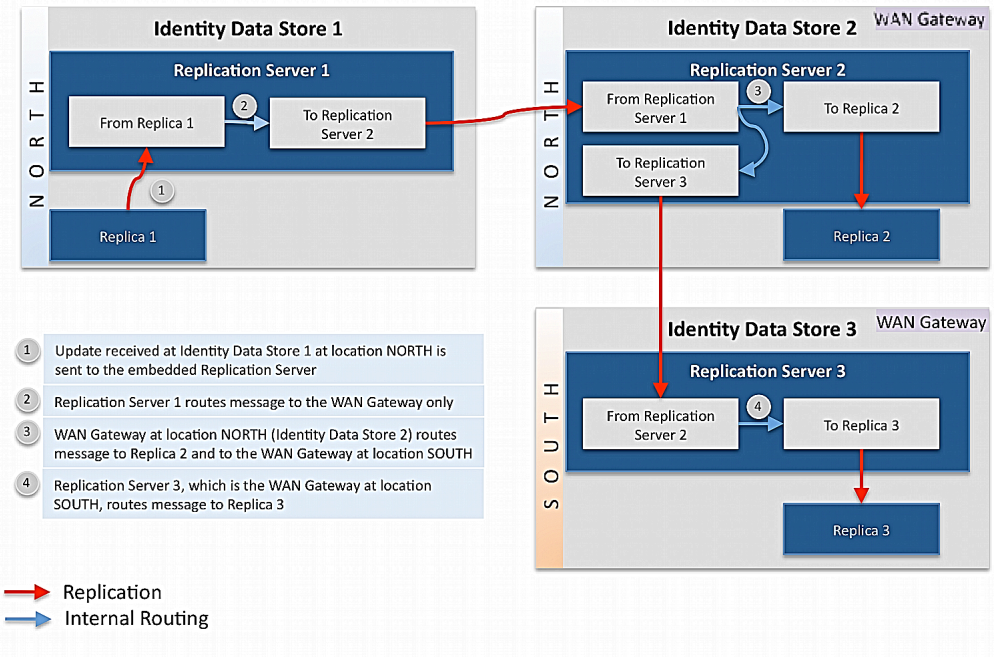
MY TITLE WAN Gateway Servers

If the WAN Gateway Server is temporarily unavailable due to a planned or unplanned downtime, the system will dynamically re-route updates to a newly designated WAN Gateway Server in the same location. The replication server with the next highest WAN Gateway priority number automatically assumes the WAN Gateway role. For deployments with entry-balancing Directory Proxy Servers, there will be one WAN Gateway Server per data set.

By default, all servers are enabled to serve as WAN Gateways and all are set to priority 5, which is simply a way to make them all equal. If necessary, the WAN Gateway priority can be changed using **`dsconfig`** after replication has been enabled.

WAN message routing

Non-gateway replication servers forward update messages from replicas to co-located replication servers only. It is the responsibility of the WAN gateway server to forward these messages to WAN gateway servers at other locations. The figure below illustrates how an update message is routed from a non-gateway server to a remote location.



MY TITLE WAN Message Routing

WAN Gateway Server selection

The WAN Gateway role is dynamically assigned to the most suitable server in a particular location. In most cases, the replication server with the higher WAN Gateway priority (e.g., priority 1 indicates the highest priority) assumes this role.

A replication server will not attempt to become a WAN Gateway if any of the following conditions exists:

- The WAN Gateway priority is set to 0
- The replication server is backlogged at server startup
- The current WAN Gateway has higher priority
- The WAN Gateway has not been elected yet, but higher priority replication servers are present in the location

The currently active WAN Gateway server will give up the gateway role in the following cases:

- The server has started the shutdown process
- The server is preparing for a scheduled maintenance cycle
- The server learns about a higher priority replication server in its location

Replication servers send WAN Gateway information to other servers at regular intervals using the replication protocol. This allows the replication servers to take the appropriate action, for example, to become a WAN Gateway, if necessary without any manual administrative action.

WAN replication in mixed-version environments

WAN Gateway Role assignment is only available on PingDirectory Servers version 3.5 or later. Legacy servers (i.e., pre-3.5) will never be selected as WAN Gateways. Updates from legacy replication servers are forwarded to all replication servers and not funneled to a single WAN Gateway. Also, updates from remote WAN Gateways will be forwarded to the legacy replication servers. Both of these actions will effectively increase WAN traffic during replication.

Recovering a replication changelog

About this task
In the event that the replication changelog is compromised (`<server-root>/changelogDb`), possibly due to a disk or NAS failure, perform the following steps.

Steps

1. Stop the server.
2. Backup **`replicationChanges`** from a remote server with the following command:

```
$ bin/backup --backupDirectory /app/backups/replicationChanges \
  --backendID replicationChanges
```

3. Copy the **`replicationChanges`** backup from the remote server and restore it on the local host as follows:

```
$ bin/restore --backupDirectory /app/tmp/replicationChanges
```

4. Start the server.

Disaster recovery

About this task
In the event that data is compromised across all systems and a restore is necessary, perform the following steps. These steps assume that no read or write operations are performed by any servers during this process.

> ⓘ **Note:**
>
> The following should be considered for disaster recovery:
>
> - With the default configuration, the server automatically exports all data nightly using the **`Export All Non-Administrative Backends`** recurring task. Up to seven days of exports are maintained. It is recommended that these be archived on another system.
> - The Data Recovery Log logs all changes in a reversible format to `logs/data-recovery/data-recovery`.
> - The **`bin/extract-data-recovery-log-changes`** tool provides the ability to redo or undo any changes from the `logs/data-recovery/data-recovery` logs.
> - The combination of these allows you to either rebuild the data set to any point in time or to revert specific changes on a live data set (for example, if an errant application mistakenly wipes out some data).

> ⓘ **Note:**
>
> See *LDIF export as a recurring task* on page 309 for information about LDIF exports.

Steps

1. Stop all servers.
2. Go to one of the servers and remove it from the topology:
   `bin/remove-defunct-server`

   This also cleans replication artifacts on this server instance as long as bind credentials are not provided. For this reason, don't provide `--bindDN` or `--bindPassword` with this.

3. The server might ask you to reconnect for each offline server, enter `no`.
4. Locate the backup or exported LDIF file that represents the last working copy of the database.
5. Restore the backup or import the LDIF file on a single server. If importing an LDIF file, use the `--excludeReplication` option with the **`bin/import-ldif`** command.
6. Start the restored server. The server can now receive client requests.
7. The server might ask you to reconnect for each offline server, enter `no`.
8. Clean up replication artifacts from the next server before starting it up:
   `bin/remove-defunct-server`
9. Start the server in lockdown mode with the following command:
   `bin/start-server --skipPrime --lockdownMode`
10. Enable replication from the first server to the second server.
    `bin/dsreplication enable`
11. Initialize the second server from the first with the following command:
    `bin/dsreplication initialize`
12. Restart the second server or use the **`bin/leave-lockdown-mode`** command to exit lockdown mode.

    The second server can now receive client requests.
13. Repeat steps 6 through 10 for any other servers.

Managing Logging

The PingDirectory Server supports a rich set of log publishers to monitor access, debug, and error messages that occur during normal server processing. Administrators can view the standard set of default log files as well as configure custom log publishers with pre-defined filtering with its own log rotation and retention policies.

Default Directory Server logs

The Directory Server provides a standard set of default log files to monitor the server activity. You can view this set of logs in the `PingDirectory/logs` directory. The following default log files are available on the Directory Server and are presented below.

Directory Server Logs

Log File	Description
access	File-based Access Log that records operations processed by the Directory Server. Access log records can be used to provide information about problems during operation processing (for example, unindexed searches, failed requests, etc.), and provide information about the time required to process each operation.
change-notifications.log	Records changes to data anywhere in the server, which match one or more configured change subscriptions.
config-audit.log	Records information about changes made to the Directory Server configuration in a format that can be replayed using the **`dsconfig`** tool
errors	File-based Error Log. Provides information about warnings, errors, and significant events that occur during server processing.
expensive-ops	Expensive Operations Log. Disabled by default. Provides only those operations that took longer than 1000 milliseconds to complete.
failed-ops	Failed Operations Log. Provides information on all operations that failed in single-line log format.

Log File	Description
replication	Records any replication errors and publishes them to the file system.
searches-returning-no-entries	Searches Returning No Entries Log. Disabled by default. Provides information only on operations that did not return any entries.
server.out	Records anything written to standard output or standard error, which includes startup messages. If garbage collection debugging is enabled, then the information will be written to server.out.
server.pid	Stores the server's process ID.
server.status	Stores the timestamp, a status code, and an optional message providing additional information on the server status.
setup.log	Records messages that occur during the initial configuration of a Directory Server with the **`setup`** tool.
tools	Directory that holds logs for long running utilities: **`import-ldif`**, **`export-ldif`**, **`backup`**, **`restore`**, **`verify-index`**. Current and previous copies of the log are present in the Directory Server.
update.log	Records messages that occur during a Directory Server update.

Types of log publishers

The PingDirectory Server provides several classes of log publishers for parsing, aggregating, and filtering information events that occur during normal processing in the server. There are three primary types of Log Publishers: access logs, debug logs, and error logs. Each type has multiple subtypes of log files based on the log server type:

Types of Log Publishers

Log Publisher Type	Description	
Access	Provides the requests received and responses returned by the Directory Server. The information can be used to understand the operations performed by clients and to debug problems with the client applications. It can also be used for collecting usage information for performance and capacity planning purposes. There are tools described later that can analyze the access log to provide summaries of the LDAP activity and performance.	
	File-based Audit Log	Special type of access logger that provides detailed information about changes processed within the server. Disabled by default. ▪ **Data Recovery Log**. Publishes information about each write operation processed in the server in a manner that allows those changes to be reverted or replayed (in conjunction with the extract-data-recovery-log-changes tool) if data recovery is needed.
	JDBC-based Access Log	Stores access log information using a JDBC database connection. Disabled by default.

Log Publisher Type	Description	
	File-based Access Logs	Provides a character-based stream used by TextWriter Publishers as a target for outputting log records. There are six types of file-based loggers: ▪ **Admin Alert Access Log**. Generates administrative alerts for any operations that match the criteria for this access logger. Disabled by default. ▪ **File-based Access Log**. Publishes access log messages to the file system. Enabled by default. ▪ **Syslog-based Access Log**. Publishes access log messages to a syslogd port. Disabled by default. ▪ **Expensive-Operations Access Log**. Publishes only those access log messages of operations that take longer than 1000 milliseconds. Disabled by default. ▪ **Failed-Operations Access Log**. Publishes only those access log messages of operations that failed for any reason. Enabled by default. ▪ **Successful Searches with No Entries Returned Log**. Publishes only those access log messages of search operations that failed to return any entries. Disabled by default. ▪ **JSON Access Logger**. Publishes JSON-formatted access log messages to the file system. ▪ **Console JSON Access Logger**. Publishes JSON-formatted access log messages to the JVM's standard output or standard error stream. This log publisher is only recommended when the server is run in no-detach mode and is best suited for use in Docker or other containers that can capture log content written to standard output or standard error. ▪ **Third-Party Access Log Publisher**. Publishes access log messages using a custom log publisher created using the Server SDK. ▪ **Third-Party File-Based Access Log Publisher**. Publishes access log messages to a file using a custom log publisher created using the Server SDK, including enhanced support for log file rotation and retention.
	HTTP Operation Log Publisherrs	▪ **Common Log File HTTP Operation Log Publisher**. Publishes information about HTTP requests in the W3C common log format. ▪ **HTTP Detailed Access**. Publishes detailed information about HTTP requests processed by the server.
Debug	Provides information about warnings, errors, or significant events that occur within the server.	

Log Publisher Type	Description	
	Debug Loggers	▪ **Debug ACI Logger**. Stores debug information on ACI evaluation for any request operations against the server. ▪ **Server SDK Extension Debug Logger** — Provides simplified access to debug messages generated by Server SDK extension
	File-based Error Logs	There are two types of File-based Error Logs: ▪ **Error log**. Publishes error messages to the file system. Enabled by default. ▪ **Replication log**. Publishes replication error messages to the file system. Enabled by default. ▪ **JSON Error Logger**. Publishes JSON-formatted error log messages to the file system. ▪ **Console JSON Access Logger**. Publishes JSON-formatted access log messages to the JVM's standard output or standard error stream. This log publisher is only recommended when the server is run in no-detach mode and is best suited for use in Docker or other containers that can capture log content written to standard output or standard error. ▪ **Third-Party Error Log Publisher**. Publishes error log messages using a custom log publisher created using the Server SDK. ▪ **Third-Party File-Based Error Log Publisher**. Publishes error log messages to a file using a custom log publisher created using the Server SDK, including enhanced support for log file rotation and retention.
	JDBC-based Error Logs	Stores error log information using a JDBC database connection. Disabled by default.
	Syslog-based Error Logs	Publishes error messages to a syslogd port.

Viewing the List of Log Publishers

You can quickly view the list of log publishers on the Directory Server using the **`dsconfig`** tool.

ⓘ **Note:** Initially, the JDBC, syslog, and Admin Alert log publishers must specifically be configured using **`dsconfig`** before they appear in the list of log publishers. Procedures to configure these types of log publishers appear later in this chapter.

To View the List of Log Publishers

Steps

- Use **`dsconfig`** to view the log publishers.

```
$ bin/dsconfig list-log-publishers
```

```
Log Publisher                                : Type              : enabled
---------------------------------------------:-------------------:--------
Debug ACI Logger                             : debug-access      : false
```

```
Expensive Operations Access Logger           : file-based-access : false
Failed Operations Access Logger              : file-based-access : true
File-Based Access Logger                     : file-based-access : true
File-Based Audit Logger                      : file-based-audit  : false
File-Based Debug Logger                      : file-based-debug  : false
File-Based Error Logger                      : file-based-error  : true
Replication Repair Logger                    : file-based-error  : true
Successful Searches with No Entries Returned : file-based-access : false
```

Enabling or Disabling a Default Log Publisher

You can enable or disable any log publisher available on the Directory Server using the **dsconfig** tool. By default, the following loggers are disabled and should be enabled only when troubleshooting an issue on the server:

- Expensive Operations Access Logger
- File-Based Audit Logger
- File-Based Debug Logger
- Successful Searches with No Entries Returned

To Enable a Default Access Log

Steps

- Use **dsconfig** to enable an Access Log Publisher. In this example, enable the Expensive-Ops log, which will record only those access log messages that take 1000 milliseconds or longer.

```
$ bin/dsconfig set-log-publisher-prop \
  --publisher-name "Expensive Operations Access Logger" --set enabled:true
```

Managing access and error log publishers

The Access Log records every request received and response returned by the Directory Server. The Access Log stores the IDs for the client connection, operation, the LDAP message involved with each client request, and the server response. The information can be used to debug any problems with a client application by correlating the numeric operation identifier to the client request or response.

The Directory Server supports multiple classes of access log publishers depending on your logging requirements. The following types of access log publishers are available on the system:

- **File-Based Access Log Publishers**. Provides a character-based TextWriter stream for outputting log records. There are three subclasses of TextWriter access logs:
  - **File-Based Access Logs**. Enabled by default. The File-based Access Log publishes access messages to the file system as `<server-root>/logs/access`. The Failed-Operations Log, Expensive-Operations Log, and the Searches with No Entries Returned Log are specialized types of the File-Based Access Log and shows only specific information necessary for troubleshooting purposes.
  - **Admin-Alert Access Logs**. Disabled by default. The Admin-Alert Access Log is specialized type of logger that automatically generates administrative alerts for any operations that match a criteria for this access log publisher.
  - **Syslog-Based Access Logs**. Disabled by default. The Syslog Access Log publishes access messages to a syslogd port.
- **File-Based Audit Logs**. Disabled by default. The Audit Log provides detailed information about modifications (writes) processed within the Directory Server. The File-based Audit Log publishes access messages to the file system as `<server-root>/logs/audit`.
- **JDBC-Based Access Logs**. Disabled by default. The JDBC-based Access Log provides information using a JDBC database connection.
- **JDBC-Based Error Logs**. Disabled by default. The JDBC-based Error Log provides information using a JDBC database connection.

Managing file-based access log publishers

The PingDirectory Server supports a flexible and configurable Access Logging system that provides a full set of customized components to meet your system's logging requirements. The default Access Log can be configured to write information to a log file with two records per operation, one for the request received and one for the response returned. It can also be configured to write one message per operation or configured to record information about a subset of operations processed by the server. In addition to modifying existing default log files, you can create custom log publishers to monitor specific properties or connection criteria. For more information, see *Creating New Log Publishers*.

The Directory Server can be configured to use multiple access log publishers writing logs to different files using different configurations. This approach makes it possible to have fine-grained logging for various purposes (for example, a log that contains only failed operations, a log that contains only operations requested by root users, or a log that contains only operations that took longer than 20ms to complete).

The Directory Server provides an additional mechanism to filter access logs to record only a subset of messages of one or more types. The access log filtering mechanism uses the operation criteria (connection, request, result, search-entry, search-reference) to determine whether a given message should be logged based on information associated with the client connection as well as information in the operation request/response messages. For more information, see *Configuring Filtered Logging*.

Access log format

The Access Log has a standard format that lists various elements identifying the connection and operation occurring within the Directory Server. By default, each operation generates one access log message.

The Access Log displays the following common properties:

- **Timestamp**. Displays the date and time of the operation. Format: DD/Month/ YYYY:HH:MM:SS <offset from UTC time>
- **Connection Type**. Displays the connection type requested by the client and the response by the server. Examples include the following:
  - CONNECT
  - BIND REQUEST/RESULT
  - UNBIND REQUEST
  - DISCONNECT
  - SEARCH REQUEST/RESULT
  - MODIFY REQUEST/RESPONSE
  - others include: ABANDON, ADD, COMPARE, DELETE, EXTENDED OPERATION, MODIFY, MODIFY DN
- **Connection ID**. Numeric identifier, starting incrementally with 0, that identifies the client connection that is requesting the operation. The connection ID is unique for a span of time on a single server. Values of the connection ID will be re-used when the server restarts or when it has had enough connections to cause the identifier to wrap back to zero.
- **Operation ID**. Numeric identifier, starting incrementally with 0, that identifies the operation. The operation ID is unique for a span of time on a single server. Values of the operation ID will be re-used when the server restarts or when it has serviced enough operations to cause the identifier to wrap back to zero.
- **Result Code**. LDAP result code that determines the success or failure of the operation result. Result messages include a result element that indicates whether the operation was successful or if failed, the general category for the failure, and an etime element that indicates the length of time in milliseconds that the server spent processing the operation.

  The Directory Server provides a useful tool **`<server-root>/bin/ldap-result-code`** (UNIX, Linux) or **`<server-root>\bat\ldap-result-code`** (Windows), that displays all of the result codes used in

the system. You can use the utility if you are not sure what a result code means. For example, use the following:

- **`ldap-result-code`** `--list` displays all of the defined result codes in the Directory Server.
- **`ldap-result-code`** `--int-value 16654` displays the name of the result code with a numeric value of 16654.
- **`ldap-result-code`** `--search operation` displays a list of all result codes whose name includes the substring "operation".

- **Elapsed Time**. Displays the elapsed time (milliseconds) during which the operation completed its processing.
- **Message ID**. Numeric identifier, starting incrementally with 1, which identifies the LDAP message used to request the operation.

Access log example

The following example shows output from the Access Log in `<server-root>/logs/access`:

```
[01/Jun/2011:14:48:17 -0500] CONNECT conn=0 from="10.8.1.243" to="10.8.1.243"
 protocol="LDAP"
[01/Jun/2011:14:48:17 -0500] BIND REQUEST conn=0 op=0 msgID=1 version="3"
 dn="cn=Directory Manager"
  authType="SIMPLE"
[01/Jun/2011:14:48:17 -0500] BIND RESULT conn=0 op=0 msgID=1 resultCode=0
 etime=26.357
  authDN="cn=Directory Manager,cn=Root DNs,cn=config"
[01/Jun/2011:14:48:17 -0500] UNBIND REQUEST conn=0 op=1 msgID=2
[01/Jun/2011:14:48:17 -0500] DISCONNECT conn=0 reason="Client Unbind"
... (more output) ...
```

Modifying the access log using dsconfig interactive mode

About this task

The File-Based Access Log can be modified to include or exclude all log messages of a given type using the **`dsconfig`** tool in interactive or non-interactive mode.

Steps

1. Use **`dsconfig`** in interactive mode to modify the access log properties.

   ```
   $ bin/dsconfig
   ```
2. Follow the prompts to specify the LDAP connection parameters for host name or IP address, connection type (LDAP, SSL, or StartTLS), port number, bind DN and password.
3. On the Directory Server main menu, type the number corresponding to the Log Publisher.
4. On the **Log Publisher Management** menu, enter the option to view and edit an existing log publisher.
5. On the **Log Publisher** menu, type the number corresponding to File-based Access Logger.
6. On the **File-Based Access Log Publisher** menu, type the number corresponding to the property that you want to change, and then follow the prompts.
7. Type `f` to apply the changes.

Modifying the access log using dsconfig non-interactive mode

About this task

You can use the **`dsconfig`** tool in non-interactive mode to quickly modify a log publisher property on the command line or in a script. For information on each property, see the *Directory Server Configuration Reference Guide*, which is an HTML document listing the various properties for each Directory Server component.

Steps

- Use **dsconfig** with the `--no-prompt` option with the properties that you want to modify or set for your access log. In this example, enable the properties to include the instance name and startup ID.

```
$ bin/dsconfig set-log-publisher-prop --publisher-name "File-Based Access
 Logger" \
  --set include-instance-name:true --set include-startup-id:true
```

Modifying the maximum length of log message strings

About this task

By default, the Directory Server sets the maximum length of log message strings to 2000 characters. This value is configurable for any access log publisher, except the Syslog publisher, which is set to 500 characters. You can change the maximum length of log message strings by setting the `max-string-length` configuration property. If any string has more than the configured number of characters, then that string will be truncated and a placeholder will be appended to indicate the number of remaining characters in the original string.

Steps

- Use **dsconfig** to set the `max-string-length` property for an access log. The following command configures the "File-based Access Logger" to include the instance name and the maximum length of the log message strings to 5000 characters.

```
$ bin/dsconfig set-log-publisher-prop \
  --publisher-name "File-Based Access Logger" \
  --set include-instance-name:true \
  --set max-string-length:5000
```

Disabling logging of inter-server periodic search requests

About this task

To disable the logging of inter-server periodic search requests and their results, perform the following steps:

Steps

1. Create a request criteria that matches all search requests with the inter-server request control.

```
dsconfig create-request-criteria --criteria-name "Inter-server Searches" \
        --type simple --set operation-type:search \
        --set any-included-request-control:1.3.6.1.4.1.30221.2.5.42
```

2. Create a request criteria that matches all requests that are not searches with the inter-server request control.

```
dsconfig create-request-criteria --criteria-name "Exclude Inter-server
 Searches" \
        --type aggregate --set "none-included-request-criteria:Inter-server
 Searches"
```

3. Configure the access logger to use the request criteria you created. This will disable the logging of internal search requests and their corresponding results to the access log.

```
dsconfig set-log-publisher-prop --publisher-name "File-Based Access Logger"
 \
        --set "request-criteria:Exclude Inter-server Searches"
```

Generating access logs summaries

About this task

The Directory Server provides a convenience tool, **summarize-access-log**, that generates a summary of one or more file-based access logs. The summary provides analytical metric information that could be useful for administrators. The following metrics are provided in each summary:

- Total time span covered by the log files.
- Number of connections established and the average rate of new connections per second.
- IP addresses of up to the top 20 of the clients that most frequently connect to the server, the number of connections by each address, and the percentage of connections of each.
- Total number of operations processed, the number of operations of each type, the percentage of each type out of the total number, and the average rate per second for each type of operation.
- Average processing time for all operations and for each type of operation.
- Histogram of the processing times for each type of operation.
- Up to the 20 most common result codes for each type of operation, the number of operations of that type with that result code, and the percentage of operations of that type with that result code.
- Number of unindexed searches processed by the server.
- Breakdown of the scopes used for search operations with the number of percentage of searches with each scope.
- Breakdown of the most common entry counts for search operations with the number and percentage of searches that returned that number of entries.
- Breakdown of the most commonly used filter types for searches with a scope other than "base" (that is, those searches for which the server will use es when processing the filter). These filters will be represented in a generic manner so that any distinct assertion values or substring assertion elements will be replaced with question marks and attribute names in all lowercase characters (for example, `(givenName=John)` would be represented as `(givenName=?)`).

Steps

- Use the **bin/summarize-access-log** with path to one or more access log files.

```
$ bin/summarize-access-log /path/to/logs/access
```

```
Examining access log /path/to/logs/access Examined 500 lines in 1 file
 covering a total duration of 1 day, 22 hours, 57 minutes, 31 seconds

Total connections established: 69 (0.000/second)
Total disconnects: 69 (0.000/second)

Most common client addresses:
127.0.0.1: 61 (88.406)
10.8.1.209: 8 (11.594)

Total operations examined: 181 ...
(metric for each operation examined) ...

Average operation processing duration: 22.727ms
Average add operation processing duration: 226.600ms
Average bind operation processing duration: 5.721ms
Average delete operation processing duration: 77.692ms
Average modify operation processing duration: 35.530ms
Average search operation processing duration: 4.017ms

Count of add operations by processing time:
... (histogram for add operations) ...

Count of bind operations by processing time:
```

```
... (histogram for bind operations) ...

Count of delete operations by processing time:
... (histogram for delete operations) ...

Count of modify operations by processing time:
... (histogram for modify operations) ...

Count of search operations by processing time:
... (histogram for search operations) ...

Most common add operation result codes:
success: 11 (84.615%)
entry already exists: 2 (15.385%)

Most common bind operation result codes:
success: 4 (50.000%)
invalid credentials: 4 (50.000%)

Most common delete operation result codes:
success: 1 (100.000%)

Most common modify operation result codes:
success: 9 (69.231%)
no such object: 4 (30.769%)

Most common search operation result codes:
success: 133 (91.724%)
no such object: 12 (8.276%)

Number of unindexed searches: 0

Most common search scopes:
BASE: 114 (78.621%)
SUB: 16 (11.034%)
ONE: 15 (10.345%)

Most common search entry counts:
1: 119 (82.069%)
0: 17 (11.724%)
2: 5 (3.448%)
10: 4 (2.759%)

Most common generic filters for searches with a non-base scope:
(objectclass=?): 19 (61.290%)
(ds-backend-id=?): 12 (38.710%)
```

About log compression

The Directory Server supports the ability to compress log files as they are written. This feature can significantly increase the amount of data that can be stored in a given amount of space, so that log information can be kept for a longer period of time.

Because of the inherent problems with mixing compressed and uncompressed data, compression can only be enabled at the time the logger is created. Compression cannot be turned on or off once the logger is configured. Further, because of problems in trying to append to an existing compressed file, if the server encounters an existing log file at startup, it will rotate that file and begin a new one rather than attempting to append to the previous file.

Compression is performed using the standard gzip algorithm, so compressed log files can be accessed using readily-available tools. The **`summarize-access-log`** tool can also work directly on compressed log files, rather than requiring them to be uncompressed first. However, because it can be useful to have a small amount of uncompressed log data available for troubleshooting purposes, administrators using

compressed logging may wish to have a second logger defined that does not use compression and has rotation and retention policies that will minimize the amount of space consumed by those logs, while still making them useful for diagnostic purposes without the need to uncompress the files before examining them.

You can configure compression by setting the `compression-mechanism` property to have the value of "gzip" when creating a new logger.

About log signing

The Directory Server supports the ability to cryptographically sign a log to ensure that it has not been modified in any way. For example, financial institutions require audit logs for all transactions to check for correctness. Tamper-proof files are therefore needed to ensure that these transactions can be properly validated and ensure that they have not been modified by any third-party entity or internally by unscrupulous employees. You can use the **`dsconfig`** tool to enable the `sign-log` property on a Log Publisher to turn on cryptographic signing.

When enabling signing for a logger that already exists and was enabled without signing, the first log file will not be completely verifiable because it still contains unsigned content from before signing was enabled. Only log files whose entire content was written with signing enabled will be considered completely valid. For the same reason, if a log file is still open for writing, then signature validation will not indicate that the log is completely valid because the log will not include the necessary "end signed content" indicator at the end of the file.

To validate log file signatures, use the **`validate-file-signature`** tool provided in the `bin` directory of the server (or the `bat` directory for Windows systems).

Once you have enabled this property, you must disable and then re-enable the Log Publisher for the changes to take effect.

About encrypting log files

The Directory Server supports the ability to encrypt log files as they are written. The **`encrypt-log`** configuration property controls whether encryption will be enabled for the logger. Enabling encryption causes the log file to have an **`.encrypted`** extension (and if both encryption and compression are enabled, the extension will be **`.gz.encrypted`**). Any change that affects the name used for the log file could prevent older files from getting properly cleaned up.

Like compression, encryption can only be enabled when the logger is created. Encryption cannot be turned on or off once the logger is configured. For any log file that is encrypted, enabling compression is also recommended to reduce the amount of data that needs to be encrypted. This will also reduce the overall size of the log file. The **`encrypt-file`** tool (or custom code, using the LDAP SDK's **`com.unboundid.util.PassphraseEncryptedInputStream`**) is used to access the encrypted data.

To enable encryption, at least one encryption settings definition must be defined in the server. Use the one created during setup, or create a new one with the **`encryption-settings create`** command. By default, the encryption will be performed with the server's preferred encryption settings definition. To explicitly specify which definition should be used for the encryption, the **`encryption-settings-definition-id`** property can be set with the ID of that definition. It is recommended that the encryption settings definition is created from a passphrase so that the file can be decrypted by providing that passphrase, even if the original encryption settings definition is no longer available. A randomly generated encryption settings definition can also be created, but the log file can only be decrypted using a server instance that has that encryption settings definition.

When using encrypted logging, a small amount of data may remain in an in-memory buffer until the log file is closed. The encryption is performed using a block cipher, and it cannot write an incomplete block of data until the file is closed. This is not an issue for any log file that is not being actively written. To examine the contents of a log file that is being actively written, use the **`rotate-log`** tool to force the file to be rotated before attempting to examine it.

Configuring log signing

Steps

1. Use **dsconfig** to enable log signing for a Log Publisher. In this example, set the `sign-log` property on the File-based Audit Log Publisher.

```
$ bin/dsconfig set-log-publisher-prop --publisher-name "File-Based Audit
 Logger" \
  --set sign-log:true
```

2. Disable and then re-enable the Log Publisher for the change to take effect.

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

Validating a signed file

About this task

The Directory Server provides a tool, **validate-file-signature**, that checks if a file has not been tampered with in any way.

Steps

- Run the **validate-file-signature** tool to check if a signed file has been tampered with. For this example, assume that the `sign-log` property was enabled for the File-Based Audit Log Publisher.

```
$ bin/validate-file-signature --file logs/audit

All signature information in file 'logs/audit' is valid
```

ⓘ **Note:** If any validations errors occur, you will see a message similar to the one as follows:

```
One or more signature validation errors were encountered
while validating the contents of file 'logs/audit':
* The end of the input stream was encountered without
  encountering the end of an active signature block.
  The contents of this signed block cannot be trusted
  because the signature cannot be verified
```

Configuring log file encryption

Steps

1. Use **dsconfig** to enable encryption for a Log Publisher. In this example, the File-based Access Log Publisher "Encrypted Access" is created, compression is set, and rotation and retention policies are set.

```
$ bin/dsconfig create-log-publisher-prop --publisher-name "Encrypted Access"
 \
  --type file-based-access \
  --set enabled:true \
  --set compression-mechanism:gzip \
  --set encryption-settings-definition-
id:332C846EF0DCD1D5187C1592E4C74CAD33FC1E5FC20B726CD301CDD2B3FFBC2B \
```

```
  --set encrypt-log:true \
  --set log-file:logs/encrypted-access \
  --set "rotation-policy:24 Hours Time Limit Rotation Policy" \
  --set "rotation-policy:Size Limit Rotation Policy" \
  --set "retention-policy:File Count Retention Policy" \
  --set "retention-policy:Free Disk Space Retention Policy" \
  --set "retention-policy:Size Limit Retention Policy"
```

2. To decrypt and decompress the file:

```
$ bin/encrypt-file --decrypt \
  --decompress-input \
  --input-file logs/encrypted-access.20180216040332Z.gz.encrypted \
  --output-file decrypted-access
Initializing the server's encryption framework...Done
Writing decrypted data to file '/ds/PingDirectory/decrypted-access' using a
key generated from encryption settings definition
 '332c846ef0dcd1d5187c1592e4c74cad33fc1e5fc20b726cd301cdd2b3ffbc2b'
Successfully wrote 123,456,789 bytes of decrypted data
```

Creating new log publishers

The PingDirectoryProxy Server provides customization options to help you create your own log publishers with the **dsconfig** command.

When you create a new log publisher, you must also configure the log retention and rotation policies for each new publisher. For more information, see Configuring Log Rotation and Configuring Log Retention.

Creating a new log publisher

Steps

1. Use the **dsconfig** command in non-interactive mode to create and configure the new log publisher. This example shows how to create a logger that only logs disconnect operations.

```
$ bin/dsconfig create-log-publisher \
  --type file-based-access --publisher-name "Disconnect Logger" \
  --set enabled:true \
  --set "rotation-policy:24 Hours Time Limit Rotation Policy" \
  --set "rotation-policy:Size Limit Rotation Policy" \
  --set "retention-policy:File Count Retention Policy" \
  --set log-connects:false \
  --set log-requests:false --set log-results:false \
  --set log-file:logs/disconnect.log
```

ⓘ **Note:** To configure compression on the logger, add the option to the previous command:

```
--set compression-mechanism: gzip
```

Compression cannot be disabled or turned off once configured for the logger. Therefore, careful planning is required to determine your logging requirements including log rotation and retention with regards to compressed logs.

2. If needed, view log publishers with the following command:

```
$ bin/dsconfig list-log-publishers
```

Creating a log publisher using dsconfig interactive command-line mode

Steps

1. On the command line, type **bin/dsconfig**.

2. Authenticate to the server by following the prompts.
3. On the main menu, select the option to configure the log publisher.
4. On the **Log Publisher** menu, select the option to create a new log publisher.
5. Select the Log Publisher type. In this case, select **File-Based Access Log Publisher**.
6. Type a name for the log publisher.
7. Enable it.
8. Type the path to the log file, relative to the Directory Server root. For example, `logs/disconnect.log`.
9. Select the rotation policy to use for this log publisher.
10. Select the retention policy to use for this log publisher.
11. On the Log Publisher Properties menu, select the option for `log-connects:false`, `log-disconnects:true`, `log-requests:false`, and `log-results:false`.
12. Type `f` to apply the changes.

Configuring log rotation

About this task

The Directory Proxy Server allows you to configure the log rotation policy for the server. When any rotation limit is reached, the Directory Proxy Server rotates the current log and starts a new log. If you create a new log publisher, you must configure at least one log rotation policy.

You can select the following properties:

- **Time Limit Rotation Policy**. Rotates the log based on the length of time since the last rotation. Default implementations are provided for rotation every 24 hours and every 7 days.
- **Fixed Time Rotation Policy**. Rotates the logs every day at a specified time (based on 24-hour time). The default time is 2359.
- **Size Limit Rotation Policy**. Rotates the logs when the file reaches the maximum size for each log. The default size limit is 100 MB.
- **Never Rotate Policy**. Used in a rare event that does not require log rotation.

Steps

- Use **`dsconfig`** to modify the log rotation policy for the access logger.

```
$ bin/dsconfig set-log-publisher-prop \
  --publisher-name "File-Based Access Logger" \
  --remove "rotation-policy:24 Hours Time Limit Rotation Policy" \
  --add "rotation-policy:7 Days Time Limit Rotation Policy"
```

Configuring log rotation listeners

The Directory Server provides two log file rotation listeners: the copy log file rotation listener and the summarize log file rotation listener, which can be enabled with a log publisher. Log file rotation listeners allow the server to perform a task on a log file as soon as it has been rotated out of service. Custom log file listeners can be created with the Server SDK.

The copy log file rotation listener can be used to compress and copy a recently-rotated log file to an alternate location for long-term storage. The original rotated log file will be subject to deletion by a log file retention policy, but the copy will not be automatically removed. Use the following command to create a new copy log file rotation listener:

```
$ dsconfig create-log-file-rotation-listener \
  --listener-name "Copy on Rotate" \
  --type copy \
```

```
  --set enabled:true \
  --set copy-to-directory:/path/to/archive/directory \
  --set compress-on-copy:true
```

The path specified by the **copy-to-directory** property must exist, and the file system containing that directory must have enough space to hold all of the log files that will be written there. The server automatically monitors free disk space on the target file system and generates administrative alerts if the amount of free space gets too low.

The summarize log file rotation listener invokes the **summarize-access-log** tool on a recently-rotated log file and writes its output to a file in a specified location. This provides information about the number and types of operations processed by the server, processing rates and response times, and other useful metrics. Use this with access loggers that log in a format that is compatible with the **summarize-access-log** tool, including the **file-based-access** and **operation-timing-access** logger types. Use the following command to create a new summarize log file rotation listener:

```
$ dsconfig create-log-file-rotation-listener \
  --listener-name "Summarize on Rotate" \
  --type summarize \
  --set enabled:true \
  --set output-directory:/path/to/summary/directory
```

The summary output files have the same name as the rotated log file, with an extension of **.summary**. If the **output-directory** property is specified, the summary files are written to that directory. If not specified, files are placed in the directory in which the log files are written.

As with the copy log file rotation listener, summary files are not automatically deleted. Although files are generally small in comparison to the log files themselves, make sure that enough space is available in the specified storage directory. The server automatically monitors free disk space on the file system to which the summary files are written.

Configuring log retention

About this task

The Directory Server allows you to configure the log retention policy for each log on the server. When any retention limit is reached, the Directory Server removes the oldest archived log prior to creating a new log. Log retention is only effective if you have a log rotation policy in place. If you create a new log publisher, you must configure at least one log retention policy.

- **File Count Retention Policy**. Sets the number of log files you want the Directory Server to retain. The default file count is 10 logs. If the file count is set to 1, then the log will continue to grow indefinitely without being rotated.
- **Free Disk Space Retention Policy**. Sets the minimum amount of free disk space. The default free disk space is 500 MBytes.
- **Size Limit Retention Policy**. Sets the maximum size of the combined archived logs. The default size limit is 500 MBytes.
- **Time Limit Retention Policy**. Sets the maximum length of time that rotated log files should be retained.
- **Custom Retention Policy**. Create a new retention policy that meets your Directory Server's requirements. This will require developing custom code to implement the desired log retention policy.
- **Never Delete Retention Policy**. Used in a rare event that does not require log deletion.

Steps

- Use **dsconfig** to modify the log retention policy for the access logger.

```
$ bin/dsconfig set-log-publisher-prop \
  --publisher-name "File-Based Access Logger" \
```

```
  --set "retention-policy:Free Disk Space Retention Policy"
```

Configuring filtered logging

About this task

The PingDirectory Server provides a mechanism to filter access log messages based on specific criteria. The filtered log can then be used with a custom log publisher to create and to generate your own custom logs. Adding new filtered logs and associate publishers does not change the behavior of any existing logs. For example, adding a new log that only contains operations that were unsuccessful does not result in those operations being removed from the default access log.

The following example shows how to create a set of criteria that matches any operation that did not complete successfully. It then explains how to create a custom access log publisher that logs only operations matching that criteria. Note that this log does not include messages for connects or disconnects, and only a single message is logged per operation. This message contains both the request and result details.

To run log filtering based on any operation result (for example, result code, processing time, and response controls), turn off request logging and set the `include-request-details-in-result-messages` property to TRUE. Since filtering based on the results of an operation cannot be done until the operation completes, the server has no idea whether to log the request. Therefore, it might log request messages but not log any result messages. Instead, if you can only log result messages and include request details in the result messages, then only messages for operations that match the result criteria are logged. All pertinent information about the corresponding requests are included.

Steps

1. Use the **`dsconfig`** command in non-interactive mode to create a result criteria object set to failure-result-codes, a predefined set of result codes that indicate when an operation did not complete successfully.

   ```
   $ bin/dsconfig create-result-criteria --type simple \
     --criteria-name "Failed Operations" --set result-code-criteria:failure-
   result-codes
   ```

2. Use **`dsconfig`** to create the corresponding log publisher that uses the result criteria. The log rotation and retention policies are also set with this command.

   ```
   $ bin/dsconfig create-log-publisher \
     --type file-based-access \
     --publisher-name "Filtered Failed Operations" \
     --set enabled:true \
     --set log-connects:false \
     --set log-disconnects:false \
     --set log-requests:false \
     --set "result-criteria:Failed Operations" \
     --set log-file:logs/failed-ops.log \
     --set include-request-details-in-result-messages:true \
     --set "rotation-policy:7 Days Time Limit Rotation Policy" \
     --set "retention-policy:Free Disk Space Retention Policy"
   ```

3. View the `failed-ops.log` in the `logs` directory. Verify that only information about failed operations is written to it.

Managing Admin Alert Access Logs

Admin Alert Access Logs are a specialized form of filtered log that automatically generates an administrative alert when criteria configured for the log publisher matches those messages in the access log.

About access log criteria

Configuring an Admin Alert Access Log requires that you configure the criteria for the access log messages. Each criteria can be either a Simple or an Aggregate type. The Simple type uses the set of properties for the client connection, operation request, and the contents of any operation-specific requests or results. The Aggregate type provides criteria that contains Boolean combination of other operation-specific criteria objects. For more information, see the *Ping Identity Directory Server Configuration Reference*.

The criteria can be one or more of the following:

- **Connection Criteria**. Defines sets of criteria for grouping and describing client connections based on a number of properties, including protocol, client address, connection security, and authentication state.
- **Request Criteria**. Defines sets of criteria for grouping and describing operation requests based on a number of properties, including properties for the associated client connection, the type of operation, targeted entry, request controls, target attributes, and other operation-specific terms.
- **Result Criteria**. Defines sets of criteria for grouping and describing operation results based on a number of properties, including the associated client connection and operation request, the result code, response controls, privileges missing or used, and other operation-specific terms.
- **Search Entry Criteria**. Defines sets of criteria for grouping and describing search result entries based on a number of properties, including the associated client connection and operation request, the entry location and contents, and included controls.
- **Search Reference Criteria**. Defines sets of criteria for grouping and describing search result references based on a number of properties, including the associated client connection and operation request, reference contents, and included controls.

Configuring an Admin Alert Access Log publisher

About this task

Prior to configuring an Admin Alert Access Log, you must establish an administrative alert handler in your system. For more information, see *Working with Administrative Alert Handlers*.

Steps

1. Use **dsconfig** to create a criteria object for the Admin Alert Access Log. For this example, we want to log only write operations that target user entries. The following command matches any of the specified operations whose target entry matches the filter "`(objectClass=person)`".

   If you are using the **dsconfig** tool in interactive mode, the menu items for the criteria operations are located in the Standard objects menu.

   ```
   $ bin/dsconfig create-request-criteria --type simple \
     --criteria-name "User Updates" \
     --set operation-type:add \
     --set operation-type:delete \
     --set operation-type:modify \
     --set operation-type:modify-dn \
     --set "any-included-target-entry-filter:(objectClass=person)"
   ```

2. Use **dsconfig** to create a log publisher of type `admin-alert-access`.

   ```
   $ bin/dsconfig create-log-publisher \
     --publisher-name "User Updates Admin Alert Access Log" \
     --type admin-alert-access \
     --set "request-criteria:User Updates" \
     --set include-request-details-in-result-messages:true \
     --set enabled:true
   ```

Managing the Syslog-Based Access Log Publishers

The Directory Server supports access logging using the syslog protocol that has been part of the Berkeley Software Distribution (BSD) operating systems for many years. Syslog provides a flexible, albeit simple, means to generate, store and transfer log messages that is supported on most UNIX and Linux operating systems.

The quasi-standard syslog message format cannot exceed 1 kbytes and has three important parts:

- **PRI**. Specifies the message priority based on its facility and severity. The message facility is a numeric identifier that specifies the type of log messages, such as kernel messages, mail system messages, etc. The severity is a numeric identifier that specifies the severity level of the operation that is being reported. Together, the facility and the severity determine the priority of the log message indicated by angled brackets and 1-3 digit priority number. For example, "<0>", "<13>", "<103>" are valid representations of the PRI.
- **Timestamp and Host Name**. The timestamp displays the current date and time of the log. The host name or IP address displays the source of the log.
- **Message**. Displays the actual log message.

Administrators can configure syslog to handle log messages using log priorities that are based on the message's facility and severity. This feature allows users to configure the logging system in such a way that messages with high severities can be sent to a centralized repository, while lower severity messages can be stored locally on a server.

ⓘ **Note:** Since the numeric values of the severity and facility are operating system-dependent, the central repository must only include syslog messages from compatible OS types, otherwise the meanings of the PRI field is ambiguous.

Before you begin

You will need to identify the host name and port to which you want to connect. Because the Syslog Protocol uses User Datagram Protocol (UDP) packets, we highly recommend that you use `localhost` and utilize some additional logging tools, such as `syslog-ng`. UDP is unreliable and unsecure means to transfer data packets between hosts.

Logging with syslog

The PingDirectory Server can write log messages using the syslog protocol, for both access and error logs.

This allows messages to be aggregated at the system level and potentially forwarded to a centralized system. The messages are written to syslog as they are generated, so attackers do not have a chance to alter these log messages.

If you want to use syslog-based logging, configure the server to log to a syslog server running on the local server over the loopback interface. The local syslog server can then forward the messages to a remote server over a secure connection.

Logging over TCP for improved reliability is now supported.

UDP-based communication is in the clear, so a network observer can see all of the log messages. Therefore, we recommend only using syslog to log to a local syslog server and having it forward messages to a remote server in a secure manner. TLS encryption for TCP-based communication is now optionally supported, so you can safely configure the server to log directly to a remote syslog server.

UDP does not provide any feedback in regard towhether messages are successfully delivered, but TCP does provide this feedback. When using TCP-based logging, you can optionally specify information about multiple syslog servers; if the primary syslog server becomes unavailable, the logger can fail over to an alternative syslog server.

Logging access and error log messages can be logged as JSON objects or in legacy space-delimited text format.

In addition to access and error logging over syslog, loggers that can write JSON-formatted audit and HTTP operation log messages are also provided.

Default access log severity level

All messages are logged at the syslog severity level of 6, which is `Informational: informational` messages. Note that this value is not standard across different types of UNIX or Linux systems. Please consult your particular operating system.

syslog-facility properties

When using syslog, specify a facility for the access log messages. As an advanced property, you can select a number that corresponds to the facility you wish to use. The default value for the `syslog-facility` property is 1 (one) for user-level messages. Note that these values are not standard across different types of UNIX or Linux systems. Please consult your particular operating system documentation for properties specific to that system.

Syslog Facility Properties

Facility	Description
0	kernel messages
1	user-level messages (default)
2	mail system
3	system daemons
4	security/authorization messages
5	messages generated internally by syslogd
6	line printer subsystem
7	network news subsystem
8	UUCP subsystem
9	clock daemon
10	security/authorization messages
11	FTP daemon
12	NTP subsystem
13	log audit
14	log alert
15	clock daemon
16	local use 0
17	local use 1
18	local use 2
19	local use 3
20	local use 4
21	local use 5
22	local use 6
23	local use 7

queue-size property

The maximum number of log records that can be stored in the asynchronous queue is determined by the `queue-size` property. The default queue size set is 10000, which means that the server will continuously flush messages from the queue to the log. The server does not wait for the queue to fill up before flushing to the log. Therefore, lowering this value can impact performance.

Configuring a Syslog-Based Access Log Publisher

About this task

You can configure a Syslog-based Access Log Publisher using the **`dsconfig`** tool. We recommend that you use syslog locally on localhost and use `syslog-ng` to send the syslog messages to remote syslog servers.

Because syslog implementations differ by vendor, please review your particular vendor's syslog configuration.

Steps

- Use **`dsconfig`** to create a log publisher of type syslog-based-access.

  If you are using the **`dsconfig`** tool in interactive mode, the menu item for Syslog Facility is an advanced property, which can be exposed by typing `a` (for "show advanced properties") on the Syslog-Based Access Log Publisher menu.

```
$ bin/dsconfig create-log-publisher \
--publisher-name "syslog-access" \
--type syslog-based-access \
--set syslog-facility:4 \
--set enabled:true
```

Managing the File-Based Audit Log Publishers

The Directory Server provides an audit log, a specialized version of the access log, for troubleshooting problems that may occur in the course of processing. The log records all changes to the data in LDIF format so that administrators can quickly diagnose the changes an application made to the data or replay the changes to another server for testing purposes.

The audit log does not record authentication attempts but can be used in conjunction with the access log to troubleshoot security-related issues. The audit log is disabled by default because it does adversely impact the server's write performance.

Audit log format

The audit log uses standard LDIF format, so that administrators can quickly analyze what changes occurred to the data. The audit log begins logging when enabled and should be used to debug any issues that may have occurred. Some common properties are the following:

- **Timestamp**. Displays the date and time of the operation. Format: DD/Month/ YYYY:HH:MM:SS <offset from UTC time>
- **Connection ID**. Numeric identifier, starting incrementally with 0, that identifies the client connection that is requesting the operation.
- **Operation ID**. Numeric identifier, starting incrementally with 0, that identifies the operation.
- **Modifiers Name**. Displays the DN of the user who made the change.
- **Update Time**. Records the `modifyTimestamp` operational attribute.

Audit log example

The following example shows output from the Audit Log in the `<server-root>/ logs/audit`. The first entry shows when the audit log was enabled. The second entry show changes made to a user entry.

```
# 05/Jun/2011:10:29:04 -0500; conn=0; op=55
dn: cn=File-Based Audit Logger,cn=Loggers,cn=config
changetype: modify
replace: ds-cfg-enabled
ds-cfg-enabled: true
-
replace: modifiersName
modifiersName: cn=Directory Manager,cn=Root DNs,cn=config
-
replace: modifyTimestamp
modifyTimestamp: 20131010020345.546Z

# 05/Jun/2011:10:31:20 -0500; conn=2; op=1
dn: uid=user.996,ou=People,dc=example,dc=com
changetype: modify
replace: pager
pager: +1 115 426 4748
-
replace: homePhone
homePhone: +1 407 383 4949
-
replace: modifiersName
modifiersName: cn=Directory Manager,cn=Root DNs,cn=config
-
replace: modifyTimestamp
modifyTimestamp: 20131010020345.546Z
```

Enabling the File-Based Audit Log Publisher

About this task

You can enable the File-Based Audit Log Publisher using the **dsconfig** tool. The audit log can impact the Directory Server's write performance, so enable it only when troubleshooting any issues.

Steps

- Use **dsconfig** to enable the File-Based Audit Log Publisher. For this example, the instance name and startup ID is also enabled in the audit log.

```
$ bin/dsconfig set-log-publisher-prop \
  --publisher-name "File-Based Audit Logger" \
  --set enabled:true \
  --set include-instance-name:true \
  --set include-startup-id:true
```

Obscuring values in the audit log

You can obscure the values of specific attributes in the audit log using the `obscure-attribute` property. Each value of an obscured attribute is replaced in the audit log with a string of the form "***** OBSCURED VALUE *****". By default, attributes are not obscured, because the values of password attributes appear in hashed form rather than in the clear.

Managing the JDBC Access Log Publishers

The Directory Server supports the Java Database Connectivity (JDBCTM) API, which allows access to SQL datastores by means of its JDBC drivers. The JDBC 4.0 API, part of the Java SDK, provides a seamless method to interface with various database types in heterogeneous environments.

By easily connecting to a database, the Directory Server can be configured to implement a centralized logging system with different databases. Centralized logging simplifies log correlation and analysis tasks and provides security by storing data in a single repository. Some disadvantages of centralized logging

are that data flow asymmetries may complicate synchronization or network provisioning and may unduly burden the central repository with possibly heavy loads.

Before you begin

Before configuring the JDBC Access Log Publisher, you need to carry out two essential steps to set up the database.

- Install the database drivers in the Directory Server `lib` directory.
- Define the log mapping tables needed to map access log elements to the database column data. Only those elements in the log mapping table gets logged by the JDBC Log Publisher.

The following sections provide more information about these tasks.

Configuring the JDBC drivers

About this task

The Directory Server supports a number of JDBC drivers available in the market. We highly recommend using the JDBC 4 drivers supported in the Java platform. For example, for Oracle databases, you need to use the `ojdbc.jar` driver for Java and any associated JAR files (National Language Support jars, and others) required to connect with the particular database. The following databases are supported:

- DB2
- MySQL
- Oracle Call Interface (OCI)
- Oracle Thin
- PostgreSQL
- SQL Server

Steps

- Obtain the JAR file(s) for your particular database, and copy it into the `<server-root>/lib` directory.

Configuring the log field mapping tables

About this task

The log field mapping table associates access log fields to the database column names. You can configure the log field mapping table using the **`dsconfig`** tool, which then generates a DDL file that you can import into your database. The DDL file is generated when you create the JDBC Log Publisher.

To uniquely identify a log record, we recommend always mapping the following fields: `timestamp`, `startupid`, `message-type`, `connection-id`, `operation-type`, `instance-name`.

The table name is not part of this mapping.

The Directory Server also provides three options that you can quickly select for creating a log field mapping table:

- **Complete JDBC Access Log Field Mappings**. Maps all 52 object properties.
- **Complete JDBC Error Log Field Mappings**. Maps all 8 object properties.
- **Simple JDBC Access Log Field Mappings**. Maps a common set of object properties.
- **Custom JDBC Access Log Field Mappings**. Create a custom set of JDBC log field mappings.
- **Custom JDBC Error Log Field Mappings**. Create a custom set of JDBC error log field mappings.

Steps

1. Use **`dsconfig`** to create a log field mapping table. On the main menu, type `o` to change to the Standard Object menu, and type the number corresponding to Log Field Mapping.

2. On the **Log Field Mapping management** menu, enter the option to create a new Log Field Mapping.
3. On the **Log Field Mapping template** menu, enter the option to select a complete JDBC Access Log Field mapping to use as a template for your new field mapping.
4. Next, enter a name for the new field mapping. In this example, type `my-jdbc-test`.
5. On the **Access Log Field Mapping Properties** menu, select a property for which you want to change the value. Any property that is undefined will not be logged by the JDBC Access Log Publisher. When complete, type `f` to save and apply the changes.
6. On the **Log Field Mapping Management** menu, type `q` to exit the menu.
7. View the existing Log Mappings on the system.

```
$ bin/dsconfig list-log-field-mappings
```

```
Log Field Mapping                     : Type
--------------------------------------:-------
Complete JDBC Access Log Field Mappings : access
Complete JDBC Error Log Field Mappings  : error
my-jdbc-test                          : access
Simple JDBC Access Log Field Mappings   : access
```

Configuring the JDBC Access Log Publisher using dsconfig interactive mode

About this task

After setting up the drivers and the log mapping table, use the **`dsconfig`** utility to configure the JDBC Access Log Publisher on the Directory Server. The following example uses **`dsconfig`** interactive mode to illustrate the steps required to configure the log publisher and the external database server.

Steps

1. Copy the database JAR files to the `<server-root>/lib` directory, and then restart the Directory Server.
2. Launch the **`dsconfig`** tool in interactive command-line mode.

```
$ bin/dsconfig
```

3. Next, type the connection parameters to bind to the Directory Server. Enter the host name or IP address, type of LDAP connection (LDAP, SSL, or StartTLS) that you are using on the Directory Server, the LDAP listener port number, the user bind DN, and the bind DN password.
4. On the main menu, type the number corresponding to Log Publisher.
5. On the **Log Publisher management** menu, enter the option to create a new Log Publisher.
6. On the **Log Publisher template** menu, type n to create a new Log Publisher.
7. On the **Log Publisher Type** menu, enter the option to create a new JDBC-Based Access Log Publisher.
8. Type a name for the JDBC Access Log Publisher.
9. On the **Enabled Property** menu, enter the option to enable the log publisher.
10. On the **Server Property** menu, enter the option to create a new JDBC External Server.
11. Next, type the name for the JDBC External Server. This is a symbolic name used to represent the DBMS.
12. On the **JDBC Driver Type Property** menu, type the number corresponding to the type of JDBC database driver type.
13. Next, type a name for the `database-name` property. This is the DBMS database name. The database name must contain the table referred to in the generated DDL.
14. Next, type the host name or IP address (server-host-name) of the external server.

15. Type the server listener port. In this example, type `1541`.
16. Review the properties for the external server, and the type `f` to apply the changes.
17. If you need to supply your own JDBC URL, type a for advanced properties to open the `jdbc-driver-url` property and supply the appropriate URL. The example below shows how to access an Oracle Thin Client connection using a SID instead of a Service.

```
>>>> Configure the properties of the JDBC External Server

Property Value(s)
-----------------------------------------------------
1) description -
2) jdbc-driver-type oraclethin
3) jdbc-driver-url jdbc:oracle:thin@myhost:1541:my_SID
4) database-name jdbc-test
5) server-host-name localhost
6) server-port 1541
7) user-name -
8) password -

?) help
f) finish - create the new JDBC External Server
a) hide advanced properties of the JDBC External Server
d) display the equivalent dsconfig arguments to create this object
b) back
q) quit

Enter choice [b]: f
```

```
JDBC External Server was created successfully
```

18. When the JDBC Log Publisher is created, the Directory Server automatically generates a DDL file of the Log Field Mappings in the `<server-root>/logs/ddls/<name-of-logger>.sql` file. Import the DDL file to your database.

Configuring the JDBC Access Log Publisher using dsconfig non-interactive mode

About this task

The following example uses **`dsconfig`** non-interactive mode to illustrate the steps to configure the log publisher and the external database server presented in the previous section.

Steps

1. Use **`dsconfig`** with the `--no-prompt` option to create the JDBC external server.

```
$ bin/dsconfig --no-prompt create-external-server \
  --server-name jdbc-external \ --type jdbc \
  --set jdbc-driver-type:oraclethin \
  --set database-name:ubid_access_log \
  --set server-host-name:localhost --set server-port:1541
```

2. Use **`dsconfig`** to create the log publisher.

```
$ bin/dsconfig --no-prompt create-log-publisher \
  --publisher-name jdbc-test \
  --type jdbc-based-access \
  --set enabled:true \
  --set server:jdbc-external \
  --set "log-field-mapping:Simple JDBC Access Log Field Mappings"
```

3. When the JDBC Log Publisher is created, the Directory Server automatically generates a DDL file of the Log Field Mappings in the `<server-root>/logs/ddls/<name- of-logger>.sql` file. Import the DDL file to your database.

   The procedure to configure the JDBC-Based Error Log Publisher is similar to creating a JDBC-Based Access Log Publisher. You can run the previous **`dsconfig`** command with the `--type jdbc-based-error` as follows:

```
$ bin/dsconfig --no-prompt create-log-publisher \
--publisher-name jdbc-error-test \
--type jdbc-based-error \
--set enabled:true \
--set server:jdbc-external \
--set "log-field-mapping:Simple JDBC Access Log Field Mappings"
```

Managing the File-Based Error Log Publisher

The Error Log reports errors, warnings, and informational messages about events that occur during the course of the Directory Server's operation. Each entry in the error log records the following properties (some are disabled by default and must be enabled):

- **Timestamp**. Displays the date and time of the operation. Format: DD/Month/ YYYY:HH:MM:SS <offset from UTC time>
- **Category**. Specifies the message category that is loosely based on the server components.
- **Severity**. Specifies the message severity of the event, which defines the importance of the message in terms of major errors that need to be quickly addressed. The default severity levels are: `fatal-error`, `notice`, `severe-error`, `severe-warning`.
- **Message ID**. Specifies the numeric identifier of the message.
- **Message**. Stores the error, warning, or informational message.

Error log example

The following example displays the error log for the PingDataMetrics Server. The log is enabled by default and is accessible in the `<server-root>/logs/errors` file.

```
[21/Oct/2012:05:15:23.048 -0500] category=RUNTIME_INFORMATION severity=NOTICE
msgID=20381715 msg="JVM Arguments: '-Xmx8g', '-Xms8g', '-XX:MaxNewSize=1g',
'-XX:NewSize=1g', '-XX:+UseConcMarkSweepGC', '-XX:+CMSConcurrentMTEnabled',
'-XX:+CMSParallelRemarkEnabled', '-XX:+CMSParallelSurvivorRemarkEnabled',
'-XX:+CMSScavengeBeforeRemark', '-XX:RefDiscoveryPolicy=1',
'-XX:ParallelCMSThreads=4', '-XX:CMSMaxAbortablePrecleanTime=3600000',
'-XX:CMSInitiatingOccupancyFraction=80', '-XX:+UseParNewGC', '-XX:
+UseMembar',
'-XX:+UseBiasedLocking', '-XX:+UseLargePages', '-XX:+UseCompressedOops',
'-XX:PermSize=128M', '-XX:+HeapDumpOnOutOfMemoryError',
'-Dcom.unboundid.directory.server.scriptName=setup'"
[21/Oct/2012:05:15:23.081 -0500] category=EXTENSIONS severity=NOTICE
msgID=1880555611 msg="Administrative alert type=server-starting
id=4178daee-ba3a-4be5-8e07-5ba17bf30b71
class=com.unboundid.directory.server.core.MetricsEngine
msg='The PingDataMetrics Server is starting'"
[21/Oct/2012:05:15:23.585 -0500] category=CORE severity=NOTICE
msgID=1879507338 msg="Starting group processing for backend api-users"
[21/Oct/2012:05:15:23.586 -0500] category=CORE severity=NOTICE
msgID=1879507339 msg="Completed group processing for backend api-users"
[21/Oct/2012:05:15:23.586 -0500] category=EXTENSIONS severity=NOTICE
msgID=1880555575 msg="'Group cache (2 static group(s) with 0 total
memberships and 0 unique members, 0 virtual static group(s),
1 dynamic group(s))' currently consumes 7968 bytes and can grow to a maximum
of an unknown number of bytes"
[21/Oct/2012:05:16:18.011 -0500] category=CORE severity=NOTICE
msgID=458887 msg="The PingDataMetrics Server (PingDataMetrics Server 8.1.0.0
```

```
build 20121021003738Z, R12799) has started successfully"
```

Modifying the File-Based Error Logs

Steps

- Use **dsconfig** to modify the default File-Based Error Log.

```
$ bin/dsconfig set-log-publisher-prop \
  --publisher-name "File-Based Error Logger" \
  --set include-product-name:true --set include-instance-name:true \
  --set include-startup-id:true
```

Managing the Syslog-Based Error Log Publisher

The Directory Server supports a Syslog-Based Error Log Publisher using the same mechanism as the Syslog-Based Access Log Publisher. You can easily configure the error logger using the **dsconfig** tool.

Syslog error mapping

The Directory Server automatically maps error log severities to the syslog severities. The following mappings are used:

Error to Syslog Severities Mappings to Syslog

Error Log Facility	Syslog Severity
FATAL_ERROR,0	Syslog Emergency
SEVERE_ERROR,1	Syslog Alert
SEVERE_WARNING,2	Syslog Critical
MILD_ERROR,3	Syslog Error
MILD_WARNING,4	Syslog Warn
NOTICE,5	Syslog Notice
INFORMATION,6	Syslog Info
DEBUG,7	Syslog Debug

Configuring a Syslog-Based Error Log Publisher

About this task

You can configure a Syslog-based Error Log Publisher using the **dsconfig** tool. Again, we recommend that you use syslog locally on `localhost` and use **syslog-ng** to send the data packets over the UDP protocol.

Because syslog implementations differ by vendor, please review your particular vendor's syslog configuration.

Steps

- Use **dsconfig** to create a log publisher of type `syslog-based-error`. In this example, set the syslog facility to 4 for security/authorization messages.

```
$ bin/dsconfig create-log-publisher --publisher-name "syslog-error" \
  --type syslog-based-error --set syslog-facility:4 --set enabled:true
```

Creating File-Based Debug Log Publishers

The Directory Server provides a File-Based Debug Log Publisher that can be configured when troubleshooting a problem that might occur during server processing. Because the debug data may be too large to maintain during normal operations, the Debug Log Publisher must be specifically configured and enabled. The Debug Log reports the following types of information:

- Exception data thrown by the server.
- Data read or written to network clients.
- Data read or written to the database.
- Access control or password policy data made within the server.

You can use the **dsconfig** tool to create a debug log publisher. You should only create a debug logger when troubleshooting a problem due to the voluminous output the Directory Server generates.

Creating a File-Based Debug Log Publisher

Steps

- Use **dsconfig** to create the debug log publisher. The `log-file` property (required) sets the debug log path. You must also specify the rotation and retention policy for the debug log.

```
$ bin/dsconfig create-log-publisher \
  --publisher-name "File-Based Debug Logger" \
  --type file-based-debug \
  --set enabled:true \
  --set log-file:/logs/debug \
  --set "rotation-policy:24 Hours Time Limit Rotation Policy" \
  --set "rotation-policy:Size Limit Rotation Policy" \
  --set "retention-policy:File Count Retention Policy"
```

Deleting a File-Based Debug Log Publisher

Steps

- Use **dsconfig** to create the debug log publisher.

```
$ bin/dsconfig delete-log-publisher \
  --publisher-name "File-Based Debug Logger"
```

Managing Monitoring

The PingDirectory Server also provides a flexible monitoring framework that exposes its monitoring information under the `cn=monitor` entry and provides interfaces through the PingDataMetrics Server, the Administrative Console, SNMP, JMX, and over LDAP. The Directory Server also provides a tool, the Periodic Stats Logger, to profile server performance.

The monitor backend

The Directory Server exposes its monitoring information under the `cn=monitor` entry. Administrators can use various means to monitor the servers, including the PingDataMetrics Server, through SNMP, the Administrative Console, JConsole, LDAP command-line tools, and the Periodic Stats Logger. Use the **bin/status** tool to display server component activity and state.

The list of all monitor entries can be seen using ldapsearch as follows:

```
$ bin/ldapsearch --hostname server1.example.com --port 1389 \
 --bindDN "uid=admin,dc=example,dc=com" --bindPassword secret \
 --baseDN "cn=monitor" "(objectclass=*)" cn
```

The following table describes a subset of the monitor entries:

Directory Server Monitoring Components

Component	Description
Active Operations	Provides information about the operations currently being processed by the Directory Server. Shows the number of operations, information on each operation, and the number of active persistent searches.
Backends	Provides general information about the state of an a Directory Server backend, including the entry count. If the backend is a local database, there is a corresponding database environment monitor entry with information on cache usage and on-disk size.
Client Connections	Provides information about all client connections to the Directory Server. The client connection information contains a name followed by an equal sign and a quoted value (e.g., connID="15", connectTime="20100308223038Z", etc.)
Connection Handlers	Provides information about the available connection handlers on the Directory Server, which includes the LDAP and LDIF connection handlers. These handlers are used to accept client connections and to read requests and send responses to those clients.
Disk Space Usage	Provides information about the disk space available to various components of the Directory Server.
General	Provides general information about the state of the Directory Server, including product name, vendor name, server version, etc.
Index	Provides on each index. The monitor captures the number of keys preloaded, and counters for read/write/remove/open-cursor/read-for-search. These counters provide insight into how useful an index is for a given workload.
HTTP/HTTPS Connection Handler Statistics	Provides statistics about the interaction that the associated HTTP connection handler has had with its clients, including the number of connections accepted, average requests per connection, average connection duration, total bytes returned, and average processing time by status code.
JVM Stack Trace	Provides a stack trace of all threads processing within the JVM.
LDAP Connection Handler Statistics	Provides statistics about the interaction that the associated LDAP connection handler has had with its clients, including the number of connections established and closed, bytes read and written, LDAP messages read and written, operations initiated, completed, and abandoned, etc.
Processing Time Histogram	Categorizes operation processing times into a number of user-defined buckets of information, including the total number of operations processed, overall average response time (ms), number of processing times between 0ms and 1ms, etc.
System Information	Provides general information about the system and the JVM on which the Directory Server is running, including system host name, operation system, JVM architecture, Java home, Java version, etc.
Version	Provides information about the Directory Server version, including build ID, version, revision number, etc.

Component	Description
Work Queue	Provides information about the state of the Directory Server work queue, which holds requests until they can be processed by a worker thread, including the requests rejected, current work queue size, number of worker threads, and number of busy worker threads. The work queue configuration has a **`monitor-queue-time`** property set to **`true`** by default. This logs messages for new operations with a **`qtime`** attribute included in the log messages. Its value is expressed in milliseconds and represents the length of time that operations are held in the work queue. A dedicated thread pool can be used for processing administrative operations. This thread pool enables diagnosis and corrective action if all other worker threads are processing operations. To request that operations use the administrative thread pool, using the **`ldapsearch`** command for example, use the **`--useAdministrativeSession`** option. The requester must have the **`use-admin-session`** privilege (included for root users). By default, eight threads are available for this purpose. This can be changed with the **`num-administrative-session-worker-threads`** property in the work queue configuration.

Monitoring disk space usage

The disk space usage monitor provides information about the amount of usable disk space available for Directory Server components. The disk space usage monitor evaluates the free space at locations registered through the `DiskSpaceConsumer` interface by various components of the server. Disk space monitoring excludes disk locations that do not have server components registered. However, other disk locations may still impact server performance, such as the operating system disk, if it becomes full. When relevant to the server, these locations include the server root, the location of the `/config` directory, the location of every log file, all JE backend directories, the location of the changelog, the location of the replication environment database, and the location of any server extension that registers itself with the DiskSpaceConsumer interface.

The disk space usage monitor provides the ability to generate administrative alerts, as well as take additional action if the amount of usable space drops below the defined thresholds.

Three thresholds can be configured for this monitor:

- **Low space warning threshold**. This threshold is defined as either a percentage or absolute amount of usable space. If the amount of usable space drops below this threshold, then the Directory Server will generate an administrative alert but will remain fully functional. It will generate alerts at regular intervals that you configure (such as once a day) unless action is taken to increase the amount of usable space. The Directory Server will also generate additional alerts as the amount of usable space is further reduced (e.g., each time the amount of usable space drops below a value 10% closer to the low space error threshold). If an administrator frees up disk space or adds additional capacity, then the server should automatically recognize this and stop generating alerts.
- **Low space error threshold**. This threshold is also defined as either a percentage or absolute size. Once the amount of usable space drops below this threshold, then the server will generate an alert notification and will begin rejecting all operations requested by non-root users with "UNAVAILABLE" results. The server should continue to generate alerts during this time. Once the server enters this mode, then an administrator will have to take some kind of action (e.g., running a command to invoke a task or removing a signal file) before the server will resume normal operation. This threshold must be less than or equal to the low space warning threshold. If they are equal, the server will begin rejecting requests from non-root users immediately upon detecting low usable disk space.
- **Out of space error threshold**. This threshold may also be defined as a percentage or absolute size. Once the amount of usable space drops below this threshold, then the PingDirectory Server will generate a final administrative alert and will shut itself down. This threshold must be less than or equal

to the low space error threshold. If they are equal, the server will shut itself down rather than rejecting requests from non-root users.

- **Disk space monitoring for tools**. The server monitors disk space consumption during processing for the `export-ldif`, `rebuild-index`, and `backup` tools. Space is monitored every 10 seconds if usable space for all monitored paths is greater than 15 percent of the capacity of those volumes. If usable space for any path drops below 15 percent, or below 10GB free, the space check frequency is increased to every second. Warning messages are generated if available space falls below 10 percent, or below 5GB free. If usable space for any path drops below two percent, or 1GB free, the tool processing is aborted and files may be removed to free up space.

The default configuration uses the same values for the low space error threshold and out of space error threshold. This is to prevent having the server online but rejecting requests, which will cause problems with applications trying to interact with the server. The low space warning threshold generates an alert before the problem becomes serious, well in advance of available disk space dropping to a point that it is critical.

The default values may not be suitable for all disk sizes, and should be adjusted to fit the deployment. Determining the best values should factor in the size of the disk, how big the database may become, how much space log files may consume, and how many backups will be stored.

The threshold values may be specified either as absolute sizes or as percentages of the total available disk space. All values must be specified as absolute values or as percentages. A mix of absolute values and percentages cannot be used. The low space warning threshold must be greater than or equal to the low space error threshold, the low space error threshold must be greater than or equal to the out of space error threshold, and the out of space error threshold must be greater than or equal to zero.

If the out of space error threshold is set to zero, then the server will not attempt to automatically shut itself down if it detects that usable disk space has become critically low. If the amount of usable space reaches zero, then the database will preserve its integrity but may enter a state in which it rejects all operations with an error and requires the server (or at least the affected backends) to be restarted. If the low space error threshold is also set to zero, then the server will generate periodic warnings about low available disk space but will remain fully functional for as long as possible. If all three threshold values are set to zero, then the server will not attempt to warn about or otherwise react to a lack of usable disk space.

Monitoring with the PingDataMetrics Server

The PingDataMetrics Server is an invaluable tool for collecting, aggregating and exposing historical and instantaneous data from the various Ping Identity servers in a deployment. The PingDataMetrics Server relies on a captive PostgreSQL datastore for the metrics, which it collects from internal instrumentation across the instances, replicas, and data centers in your environment. The data is available via a Monitoring API that can be used to build custom dashboards and monitoring applications to monitor the overall health of your PingData Platform system. For more information, refer to the *PingDataMetrics Server Administration Guide*.

About the collection of system monitoring data

All PingDirectory Servers have the capability to monitor the health of the server and host system they run on for diagnostic review and troubleshooting. Initially, the servers do not collect any performance data until they are prepared for monitoring by a Metrics Server using the `monitored-servers add-servers` tool or an administrator enables system health data collection for real-time inspection and querying. At a high level, all of the important server and machine metrics which can be monitored are available in the `cn=monitor` backend.

The Stats Collector plugin is the primary driver of performance data collection for LDAP, server response, replication, local JE databases, and host system machine metrics. Stats Collector configuration determines the sample and collection intervals, granularity of data (basic, extended, verbose), types of host system collection (cpu, disk, network) and what kind of data aggregation occurs for LDAP application statistics. The Stats Collector plugin ensures that a PingDataMetrics Server is able to gather all of the detailed data required for a comprehensive diagnostic review.

The Stats Collector plugin relies exclusively on entries in the `cn=monitor` backend to sample data using LDAP queries. The Stats Collector plugin is the primary driver of performance data collection for LDAP, server response, replication, local JE databases, and host system machine metrics. Stats Collector configuration determines the sample and collection intervals, granularity of data (basic, extended, verbose), types of host system collection (cpu, disk, network) and the type of data aggregation that occurs for LDAP application statistics. The Stats Collector plugin is configured with the `dsconfig` tool and collects data using LDAP queries. For example, the `--server-info:extended` option includes collection for the following:

- CPU
- JVM memory
- Memory
- Disk information
- Network information

Utilization metrics are gathered via externally invoked OS commands, such as `iostat` and `netstat`, using platform-specific arguments and version-specific output parsing.

Enabling the Host System monitor provider automatically gathers CPU and memory utilization but only optionally gathers disk and network information. Disk and network interfaces are enumerated in the configuration by device names (e.g., `eth0` or lo), and by disk device names (e.g., `sd1, sdab, sda2, scsi0`).

Monitoring key performance indicators by application

The PingDirectory Server can be configured to track many key performance metrics (for example, throughput and response-time) by the client applications requesting them. This feature is invaluable for measuring whether the Ping Identity identify infrastructure meets all of your service-level agreements (SLA) that have been defined for client applications.

When enabled, the per-application monitoring data can be accessed in the `cn=monitor` backend, the Periodic Stats Logger, and made available for collection by the Metrics Server. See the “Profiling Server Performance Using the Periodic Stats Logger” for more information on using that component. Also, see the Directory Server Configuration section of the *PingDataMetrics Server Administration Guide* for details on configuring the server to expose metrics that interest you. Tracked application information is exposed in the PingDataMetrics Server by metrics having the 'application-name' dimension. See the documentation under `docs/metrics` of the PingDataMetrics Server for information on which metrics are available with the 'application-name' dimension.

Configuring the external Servers

Before you install the PingDataMetrics Server, you need to configure the servers you will be monitoring: PingDirectory Server, PingDirectoryProxy Server, and PingDataSync Server. The PingDataMetrics Server requires all servers to be version 3.5.0 or later. See the administration guides for each product for installation instructions.

Once you have installed the Directory Server, you can use the **`dsconfig`** tool to make configuration changes for the PingDataMetrics Server. When using the **`dsconfig`** tool interactively, set the complexity level to Advanced, so that you can make all the necessary configuration changes.

Preparing the servers monitored by the PingDataMetrics Server

The Metrics Backend manages the storage of metrics and provides access to the stored blocks of metrics via LDAP. The Metrics Backend is configured to keep a maximum amount of metric history based on log retention policies. The default retention policy uses the Default Size Limit Retention Policy, Free Disk Space Retention Policy, and the File Growth Limit Policy, limiting the total disk space used to 500 MB. This amount of disk typically contains more than 24 hours of metric history, which is ample. The Directory Server keeps a metric history so that the PingDataMetrics Server can be down for a period and then catch up when it comes back online.

The following two commands create a Retention Policy that limits the number of files to 2000, and sets the Metrics Backend to flush data to a new file every 30 seconds.

```
$ bin/dsconfig create-log-retention-policy \
  --policy-name StatsCollectorRetentionPolicy \
  --type file-count --set number-of-files:2000

$ bin/dsconfig set-backend-prop \
  --backend-name metrics --set sample-flush-interval:30s \
  --set retention-policy:StatsCollectorRetentionPolicy
```

These commands configure the Metrics Backend to keep 16 hours of metric history, which consumes about 250 MB of disk, ensuring that captured metrics are available to the PingDataMetrics Server within 30 seconds of when the metric was captured. The value of the `sample-flush-interval` attribute determines the maximum delay between when a metric is captured and when it can be picked up by the PingDataMetrics Server.

The flush interval can be set between 15 seconds and 60 seconds, with longer values resulting in less processing load on the PingDataMetrics Server. However, this flush interval increases the latency between when the metric was captured and when it becomes visible in the Dashboard Application. If you change the `sample-flush-interval` attribute to 60 seconds in the example above, then the Directory Server keeps 2000 minutes of history. Because the number of metrics produced per unit of time can vary depending on the configuration, no exact formula can be used to compute how much storage is required for each hour of history. However, 20 MB per hour is a good estimate.

Configuring the Processing Time Histogram plugin

The Processing Time Histogram plugin is configured on each Directory Server and Directory Proxy Server as a set of histogram bucket ranges. When the bucket ranges for a histogram change, the PingDataMetrics Server notices the change and marks samples differently. This process allows for histograms with the same set of bucket definitions to be properly aggregated and understood when returned in a query. If different servers have different bucket definitions, then a single metric query cannot return histogram data from the servers.

You should try to keep the Processing Time Histogram bucket definitions the same on all servers. Having different definitions restricts the ability of the PingDataMetrics Server API to aggregate histogram data across servers and makes the results of a query asking "What percentage of the search requests took less than 12 milliseconds?" harder to understand.

For each server in your topology, you must set the separate-monitor-entry-per-tracked-application property of the processing time histogram plugin to true. This property must be set to expose per-application monitoring information under `cn=monitor`. When the separate-monitor-entry-per-tracked-application property is set to true, then the per-application-ldap-stats property must be set to per-application-only on the Stats Collector Plugin and vice versa.

For example, the following **`dsconfig`** command line sets the required properties of the Processing Time Histogram plugin:

```
$ bin/dsconfig set-plugin-prop --plugin-name “Processing Time Histogram” \
  --set separate-monitor-entry-per-tracked-application:true
```

The following **`dsconfig`** command line sets the per-application-ldap-stats property of the Stats Collector plugin to per-application-only:

```
$ bin/dsconfig set-plugin-prop --plugin-name “Stats Collector” \
 --set per-application-ldap-stats:per-application-only
```

Setting the connection criteria to collect SLA statistics by application

If you want to collect data about your SLAs, you need to configure connection criteria for each Service Level Agreement that you want to track. The connection criteria are used in many areas within the server. They are used by the client connection policies, but they can also be used when the server needs to perform matching based on connection-level properties, such as filtered logging. For assistance using connection criteria, contact your authorized support provider.

For example, imagine that we are interested in collecting statistics on data that is accessed by clients authenticating as the Directory Manager. We need to create connection criteria on the Directory Server that identifies any user authenticating as the Directory Manager. The connection criteria name corresponds to the application-name dimension value that clients will specify when accessing the data via the API. When you define the Connection Criteria, change the included-user-base-dn property to include the Directory Manager's full LDIF entry.

The following **dsconfig** command line creates connection criteria for the Directory Manager:

```
$ bin/dsconfig create-connection-criteria \
  --criteria-name "Directory Manager" \
  --type simple \
  --set "included-user-base-dn:cn=Directory Manager,cn=Root DNs,cn=config"
```

Proxy considerations for tracked applications

In a proxy environment, the criteria should be defined in the Directory Proxy Server since the Directory Proxy Server passes the application name through to the Directory Server in the intermediate client control. If a client of the Directory Proxy Server or Directory Server happens to use the intermediate client control, then the client name specified in the control will be used as the application name regardless of the criteria listed in the tracked-application property.

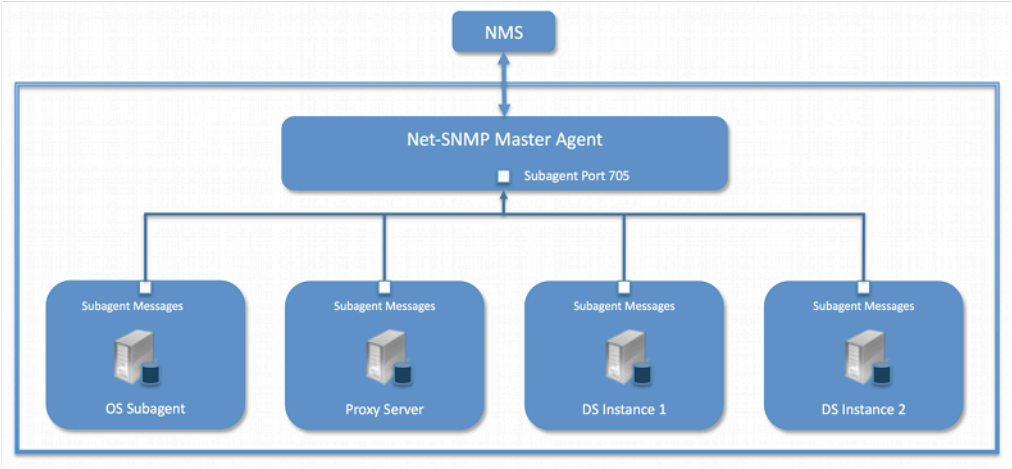
Monitoring using SNMP

The PingDirectoryProxy Server supports real-time monitoring using the Simple Network Management Protocol (SNMP). The Directory Proxy Server provides an embedded SNMPv3 subagent plugin that, when enabled, sets up the server as a managed device and exchanges monitoring information with a master agent based on the AgentX protocol.

SNMP implementation

In a typical SNMP deployment, many production environments use a network management system (NMS) for a unified monitoring and administrative view of all SNMP-enabled devices. The NMS communicates with a master agent, whose main responsibility is to translate the SNMP protocol messages and multiplex any request messages to the subagent on each managed device (for example, Directory Server instance, Directory Proxy Server, Data Sync Server, or OS Subagent). The master agent also processes responses or traps from the agents. Many vendors provide commercial NMS systems. Consult with your NMS system for specific information.

The PingDirectory Server contains an SNMP subagent plugin that connects to a Net-SNMP master agent over TCP. The main configuration properties of the plugin are the address and port of the master agent, which default to localhost and port 705, respectively. When the plugin is initialized, it creates an AgentX subagent and a managed object server, and then registers as a MIB server with the Directory Server instance. Once the plugin's startup method is called, it starts a session thread with the master agent. Whenever the connection is lost, the subagent automatically attempts to reconnect with the master agent. The Directory Server's SNMP subagent plugin transmits only read-only values for polling or trap purposes. (Set and inform operations are not supported). SNMP management applications cannot perform actions on the server on their own or by means of an NMS system.



MY TITLE Example SNMP Deployment

One important note is that the PingDirectory Server was designed to interface with a Net-SNMP (version 5.3.2.2 or later) master agent implementation with AgentX over TCP. Many operating systems provide their own Net-SNMP module. However, SMA disables some features present in the Net-SNMP package and only enables AgentX over UNIX Domain Sockets, which cannot be supported by Java. If your operating system has a native Net-SNMP master agent that only enables UNIX Domain Sockets, you must download and install a separate Net-SNMP binary from its web site.

Configuring SNMP

About this task

Because all server instances provide information for a common set of MIBs, each server instance provides its information under a unique SNMPv3 context name, equal to the server instance name. The server instance name is defined in the Global Configuration, and is constructed from the host name and the server LDAP port by default. Consequently, information must be requested using SNMPv3, specifying the context name that pertains to the desired server instance. This context name is limited to 30 characters or less. Any context name longer than 30 characters will result in an error message. Since the default context name is limited to 30 characters or less, and defaults to the server instance name and the LDAP port number, pay special attention to the length of the fully-qualified (DNS) host name.

ⓘ **Note:** The Directory Server supports SNMPv3, and only SNMPv3 can access the MIBs. For systems that implement SNMP v1 and v2c, Net-SNMP provides a proxy function to route requests in one version of SNMP to an agent using a different SNMP version.

Steps

1. Enable the Directory Server's SNMP plugin by using the **`dsconfig`** tool. Make sure to specify the address and port of the SNMP master agent. On each Directory Server instance, enable the SNMP subagent. Note that the SNMPv3 context name is limited to 30 bytes maximum. If the default dynamically-constructed instance name is greater than 30 bytes, there will be an error when attempting to enable the plugin. Enable the SNMP Subagent Alert Handler so that the sub-agent will send traps for administrative alerts generated by the server.

```
$ bin/dsconfig set-alert-handler-prop \
  --handler-name "SNMP Subagent Alert Handler" --set enabled:true
```

2. View the error log. You will see a message that the master agent is not connected, because it is not yet online.

```
The SNMP sub-agent was unable to connect to the master
agent at localhost/705: Timeout
```

3. Edit the SNMP agent configuration file, `snmpd.conf`, which is often located in `/etc/snmp/snmpd.conf`. Add the directive to run the agent as an AgentX master agent:

```
master agentx agentXSocket tcp:localhost:705
```

   Note that the use of `localhost` means that only sub-agents running on the same host can connect to the master agent. This requirement is necessary since there are no security mechanisms in the AgentX protocol.

4. Add the trap directive to send SNMPv2 traps to `localhost` with the community name, public (or whatever SNMP community has been configured for your environment) and the port.

```
trap2sink localhost public 162
```

5. To create a SNMPv3 user, add the following lines to the `/etc/snmp/snmpd.conf` file.

```
rwuser initial
createUser initial MD5 setup_passphrase DES
```

6. Run the following command to create the SNMPv3 user.

```
snmpusm -v3 -u initial -n "" -l authNoPriv -a MD5 -A setup_passphrase \
localhost create snmpuser initial
```

7. Start the **`snmpd`** daemon and after a few seconds you should see the following message in the Directory Server error log:

```
The SNMP subagent connected successfully to the master agent
at localhost:705. The SNMP context name is host.example.com:389
```

8. Set up a trap client to see the alerts that are generated by the Directory Server. Create a config file in `/tmp/snmptrapd.conf` and add the directive below to it. The directive specifies that the trap client can process traps using the public community string, and can log and trigger executable actions.

```
authcommunity log, execute public
```

9. Install the MIB definitions for the Net-SNMP client tools, usually located in the `/usr/share/snmp/mibs` directory.

```
$ cp resource/mib/* /usr/share/snmp/mibs
```

10. Then, run the trap client using the **`snmptrapd`** command. The following example specifies that the command should not create a new process using **`fork()`** from the calling shell (`-f`), do not read any configuration files (`-C`) except the one specified with the `-c` option, print to standard output (`-Lo`), and then specify that debugging output should be turned on for the User-based Security Module (`-Dusm`). The path after the `-M` option is a directory that contains the MIBs shipped with our product (i.e., `server-root/resource/mib`) .

```
$ snmptrapd -f -C -c /tmp/snmptrapd.conf -Lf /root/trap.log -Dusm \
  -m all -M +/usr/share/snmp/mibs
```

11. Run the Net-SNMP client tools to test the feature. The following options are required: `-v <SNMP version>`, `-u <user name>`, `-A <user password>`, `-l <security level>`, `-n <context name (instance name)>` . The `-m` all option loads all MIBs in the default MIB directory in `/usr/share/snmp/mibs` so that MIB names can be used in place of numeric OIDs.

```
$ snmpget -v 3 -u snmpuser -A password -l authNoPriv -n host.example.com:389
 \
-m all localhost localDBBackendCount.0
```

```
$ snmpwalk -v 3 -u snmpuser -A password -l authNoPriv -n
 host.example.com:389 \
-m all localhost systemStatus
```

MIBS

The Directory Server provides SMIv2-compliant MIB definitions (RFC 2578, 2579, 2580) for distinct monitoring statistics. These MIB definitions are to be found in text files under `resource/mib` directory under the server root directory.

Each MIB provides managed object tables for each specific SNMP management information as follows:

- **LDAP Remote Server MIB**. Provides information related to the health and status of the LDAP servers that the Directory Proxy Server connects to, and statistics about the operations invoked by the Directory Proxy Server on those LDAP servers.
- **LDAP Statistics MIB**. Provides a collection of connection-oriented performance data that is based on a connection handler in the Directory Server. A server typically contain only one connection handler and therefore supplies only one table entry.
- **Local DB Backend MIB**. Provides key metrics related to the state of the local database backends contained in the server.
- **Processing Time MIB**. Provides a collection of key performance data related to the processing time of operations broken down by several criteria but reported as a single aggregated data set.
- **Replication MIB**. Provides key metrics related to the current state of replication, which can help diagnose how much outstanding work replication may have to do.
- **System Status MIB**. Provides a set of critical metrics for determining the status and health of the system in relation to its work load.

For information on the available monitoring statistics for each MIB available on the Directory Server and the Directory Proxy Server, see the text files provided in the `resource/mib` directory below the server installation.

The Directory Server generates an extensive set of SNMP traps for event monitoring. The traps display the severity, description, name, OID, and summary. For information about the available alert types for event monitoring, see the `resource/mib/UNBOUNDID-ALERT-MIB.txt` file.

Monitoring with the Administrative Console

About this task

Ping Identity has an Administrative Console for administrators to configure the directory server. The console also provides a status option that accesses the server's monitor content.

To view the Monitor Dashboard:

Steps

1. Ensure that the Directory Server is running.
2. Open a browser to `http://server-name:8443/console`.
3. Type the root user DN and password, and then click **Login**.
4. Use the top level navigation dropdown and select 'Status.'
5. On the Administrative Console's Status page, select the Monitors tab.

Accessing the Processing Time Histogram

About this task

The PingDirectory Server provides a processing time histogram that classifies operation response time into user-defined buckets. The histogram tracks the processing on a per operation basis and as a percentage of the overall processing time for all operations. It also provides statistics for each operation type (add, bind, compare, delete, modify, modify DN, search).

Steps

1. On the Administrative Console, click **Configuration > Status > Monitors tab**.
2. Select **Processing Time Histogram**. Other monitor entries can be accessed in similar ways.

Monitoring with JMX

The PingDirectory Server supports monitoring the JVM™ through a Java Management Extensions (JMX™) management agent, which can be accessed using JConsole or any other kind of JMX client. The JMX interface provides JVM performance and resource utilization information for applications running Java. You can monitor generic metrics exposed by the JVM itself, including memory pools, threads, loaded classes, and MBeans, as well as all the monitor information that the Directory Server provides. You can also subscribe to receive JMX notifications for any administrative alerts that are generated within the server.

Running JConsole

About this task

Before you can access JConsole, you must configure and enable the JMX Connection Handler for the Directory Server using the **dsconfig** tool. See Configuring the JMX Connection Handler and Alert Handler.

To invoke the JConsole executable, type `jconsole` on the command line. If *JDK_HOME* is not set in your path, you can access JConsole in the `bin` directory of the `JDK_HOME` path.

To run JConsole:

Steps

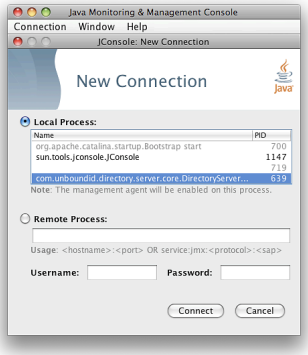
1. Use **JConsole** to open the Java Monitoring and Management Console. You can also run JConsole to monitor a specific process ID for your application: **jconsole PID**. Or you can run JConsole remotely using: **jconsole hostname:port**.

```
$ jconsole
```

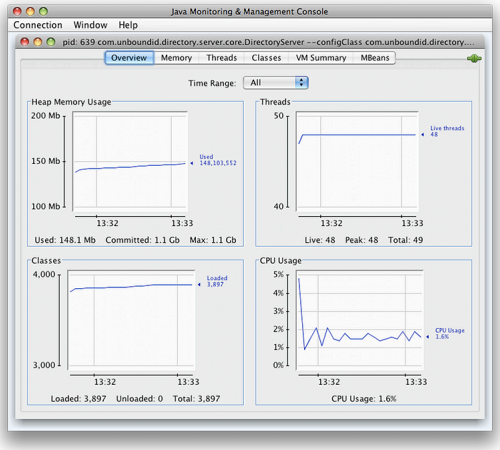
ⓘ **Note:** If SSL is configured on the JMX Connection Handler, you must specify the Directory Server jar file in the class path when running **jconsole** over SSL. For example, run the following **jconsole** command:

```
$ jconsole \
  -J-Djavax.net.ssl.trustStore=/path/to/certStores/truststore \
  -J-Djavax.net.ssl.trustStorePassword=secret \
  -J-Djava.class.path=$SERVER_ROOT/lib/PingDirectory.jar:/Library/Java/
JavaVirtualMachines/jdk-version.jdk/Contents/Home/lib/jconsole.jar
```

2. On the **Java Monitoring & Administrative Console**, click **Local Process**, and then click the **PID** corresponding to the directory server.



3. Review the resource monitoring information.



Monitoring the Directory Server using JConsole

About this task

You can set up JConsole to monitor the Directory Server using a remote process. Make sure to enable the JMX Connection Handler and to assign at least the `jmx-read` privilege to a regular user account (the `jmx-notify` privilege is required to subscribe to receive JMX notifications). Do not use a root user account, as this would pose a security risk.

Steps

1. Start the Directory Server.

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

2. Enable the JMX Connection handler using the **`dsconfig`** tool. The handler is disabled by default. Remember to include the LDAP connection parameters (host name, port, bindDN, bindPassword).

```
$ bin/dsconfig set-connection-handler-prop \
  --handler-name "JMX Connection Handler" --set enabled:true
```

3. Assign `jmx-read`, `jmx-write`, and `jmx-notify` (if the user receives notifications) to the user.

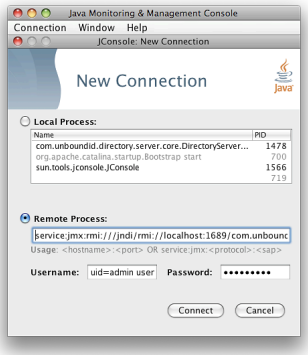
```
$ bin/ldapmodify --hostname server1.example.com --port 1389 \
  --bindDN "cn=Directory Manager" --bindPassword secret
dn: uid=admin,dc=example,dc=com
changetype: modify
```

```
replace: ds-privilege-name
ds-privilege-name: jmx-read
ds-privilege-name: jmx-write
ds-privilege-name: jmx-notify
```

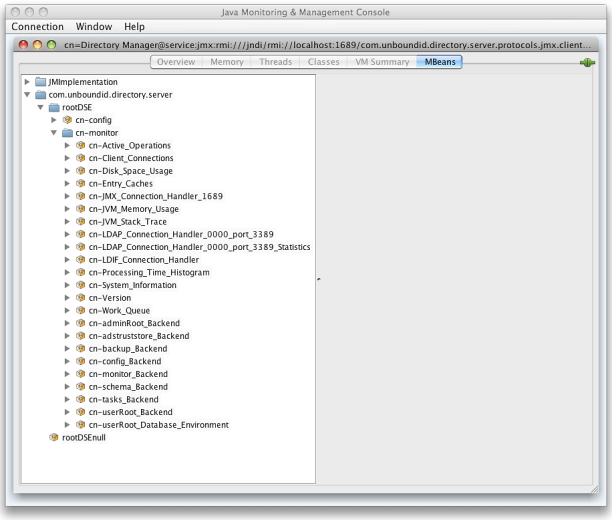
4. On the **Java Monitoring & Administrative Console**, click **Remote Process**, and enter the following JMX URL using the host and port of your Directory Server.

   ```
   service:jmx:rmi:///jndi/rmi://<host>:<port>/
   com.unboundid.directory.server.protocols.jmx.client-unknown
   ```

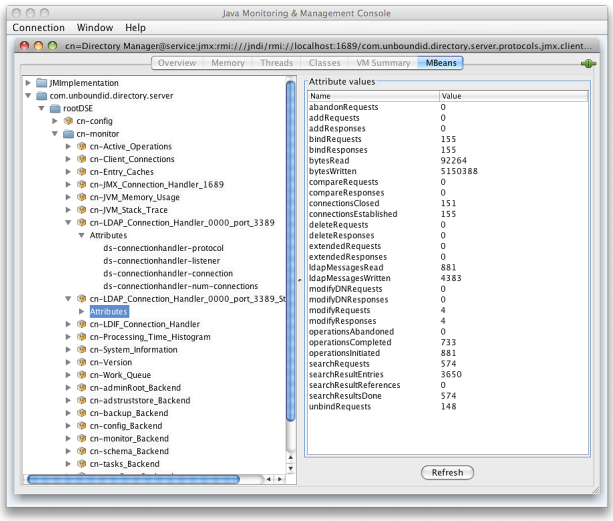
5. In the **Username** and **Password** fields, type the bind DN and password for a user that has at least the `jmx-read` privilege. Click **Connect**.



6. Click **com.unboundid.directory.server**, and expand the `rootDSE` node and the `cn-monitor` sub-node.



7. Click a monitoring entry. In this example, click the **LDAP Connection Handler** entry.



Monitoring Using the LDAP SDK

You can use the monitoring API to retrieve monitor entries from the Directory Proxy Server as well as to retrieve specific types of monitor entries.

For example, you can retrieve all monitor entries published by the Directory Server and print the information contained in each using the generic API for accessing monitor entry data as follows:

```
for (MonitorEntry e : MonitorManager.getMonitorEntries(connection))
  {
    System.out.println("Monitor Name: " + e.getMonitorName());
    System.out.println("Monitor Type: " + e.getMonitorDisplayName());
    System.out.println("Monitor Data:");
    for (MonitorAttribute a : e.getMonitorAttributes().values())
    {
      for (Object value : a.getValues())
      {
        System.out.println(" " + a.getDisplayName() + ": " +
 String.valueOf(value));
      }
     }
     System.out.println();
  }
```

For more information about the LDAP SDK and the methods in this example, see the *LDAP SDK* documentation.

Monitoring over LDAP

The PingDirectory Server exposes a majority of its information under the `cn=monitor` entry. You can access these entries over LDAP using the **`ldapsearch`** tool.

```
$ bin/ldapsearch --hostname server1.example.com --port 1389 \
  --bindDN "uid=admin,dc=example,dc=com" --bindPassword secret \
  --baseDN "cn=monitor" "(objectclass=*)"
```

Profiling Server Performance Using the Stats Logger

The Directory Proxy Server ships with a built-in Stats Logger that is useful for profiling server performance for a given configuration. At a specified interval, the Stats Logger can write server statistics to a JSON file or to a log file in a comma-separated format (.csv), which can be read by spreadsheet applications. The

logger has a negligible impact on server performance unless the `log-interval` property is set to a very small value (less than 1 second). The statistics logged and their verbosity can be customized.

The Stats Logger can also be used to view historical information about server statistics including replication, LDAP operations, host information, and gauges. Either update the configuration of the existing Stats Logger Plugin to set the advanced `gauge-info` property to `basic/extended` to include this information, or create a dedicated Periodic Stats Logger for information about statistics of interest.

For information about how to enable this logging and where to find the log file, see *To Enable the Stats Logger* on page 755.

To Enable the Stats Logger

About this task
By default, the Directory Proxy Server ships with the built-in "Stats Logger' disabled. To enable it using the **`dsconfig`** tool or the Administrative Console, go to **Plugins** menu (available on the Advanced object menu), and then, select .

Steps

1. Run **`dsconfig`** in interactive mode. Enter the LDAP or LDAPS connection parameters when prompted.

```
$ bin/dsconfig
```

2. Enter `o` to change to the Advanced Objects menu.
3. On the main menu, enter the number for Plugins.
4. On the **Plugin** menu, enter the number corresponding to view and edit an existing plugin.
5. On the **Plugin selection** list, enter the number corresponding to the Stats Logger.
6. On the **Stats Logger Plugin** menu, enter the number to set the `enabled` property to `TRUE`. When done, enter `f` to save and apply the configuration. The default logger will log information about the server every second to `<server-root>/logs/dsstats.csv`. If the server is idle, nothing will be logged, but this can be changed by setting the `suppress-if-idle` property to `FALSE` (`suppress-if-idle=false`).

```
>>>> Configure the properties of the Stats Logger Plugin

Property                Value(s)
-------------------------------------------------------------------------
1)   description        Logs performance stats to a log file
                        periodically.
2)   enabled                      false
3)   local-db-backend-info       basic
4)   replication-info             basic
5)   entry-cache-info             -
6)   host-info                    -
7)   included-ldap-application    If per-application LDAP stats is enabled,
                                  then stats will be included for all
                                  applications.
8)   log-interval                 1 s
9)   collection-interval         200 ms
10)  suppress-if-idle             true
11)  header-prefix-per-column     false
12)  empty-instead-of-zero        true
13)  lines-between-header         50
14)  included-ldap-stat           active-operations, num-connections,
                                  op-count-and-latency, work-queue
15)  included-resource-stat       memory-utilization
16)  histogram-format             count
17)  histogram-op-type            all
18)  per-application-ldap-stats   aggregate-only
```

```
19)  ldap-changelog-info          -
20)  gauge-info                   none
21)  log-file                     logs/dsstats.csv
22)  log-file-permissions         640
23)  append                       true
24)  rotation-policy              Fixed Time Rotation Policy, Size Limit
                                  Rotation Policy
25)  retention-policy             File Count Retention Policy

?)   help
f)   finish - apply any changes to the Periodic Stats Logger Plugin
a)   hide advanced properties of the Periodic Stats Logger Plugin
d)   display the equivalent dsconfig command lines to either re-create this
           object or only to apply pending changes
b)   back
q)   quit

Enter choice [b]:
```

7. Run the Directory Proxy Server. For example, if you are running in a test environment, you can run the **`search-and-mod-rate`** tool to apply some searches and modifications to the server. You can run **`search-and-mod-rate`** `--help` to see an example command.
8. View the Stats log output at `<server-root>/logs/dsstats.csv`. You can open the file in a spreadsheet.

To Configure Multiple Periodic Stats Loggers

About this task

Multiple Periodic Stats Loggers can be created to log different statistics, view historical information about gauges, or to create a log at different intervals (such as logging cumulative operations statistics every hour). To create a new log, use the existing Stats Logger as a template to get reasonable settings, including rotation and retention policy.

Steps

1. Run **`dsconfig`** by repeating steps 1–3 in *To Enable the Stats Logger*.
2. From the **Plugin management** menu, enter the number to create a new plugin.
3. From the **Create a New Periodic Stats Logger Plugin** menu, enter `t` to use an existing plugin as a template.
4. Enter the number corresponding to the existing stats logger as a template.
5. Next, enter a descriptive name for the new stats logger. For this example, type `Stats Logger-10s`.
6. Enter the log file path to the file. For this example, type `logs/dsstats2.csv`.
7. On the menu, make any other change to the logger. For this example, change the `log-interval` to 10s, and the `suppress-if-idle` to false. When finished, enter `f` to save and apply the configuration.
8. You should now see two loggers `dsstats.csv` and `dsstats2.csv` in the `logs` directory.

StatsD monitoring endpoint

The Monitoring Endpoint configuration type provides the StatsD Endpoint type that you can use to transfer metrics data in the StatsD format.

Examples of metrics you can send are:

- Busy worker thread count
- Garbage collection statistics
- Host system metrics such as CPU and memory

For a list of available metrics, use the interactive **dsconfig** menu for the Stats Collector Plugin or in the Administrative Console, edit the **Stats Collector** plugin as explained in the second example.

You configure the Monitoring Endpoint using the **dsconfig** command. When you configure Monitoring Endpoint, you include:

- The endpoint's hostname
- The endpoint's port
- A toggle to use TCP or UDP
- A toggle to use SSL if you use TCP

For example, to configure a new StatsD Monitoring Endpoint to send UDP data to localhost port 8125 using **dsconfig**:

```
dsconfig create-monitoring-endpoint \
    --type statsd \
    --endpoint-name StatsDEndpoint \
    --set enabled:true \
    --set hostname:localhost \
    --set server-port:8125 \
    --set connection-type:unencrypted-udp
```

If you are using the Administrative Console:

1. Click **Show Advanced Configuration**.
2. In the **Logging, Monitoring, and Notifications** section, click **Monitoring Endpoints**.
3. Click **New Monitoring Endpoint**.

You can send data to any number of monitoring endpoints.

The Stats Collector Plugin controls the metrics used by the StatsD monitoring endpoint. To send metrics with the StatsD monitoring endpoint, you must enable the Stats Collector Plugin. Also, you must configure the Stats Collector Plugin to indicate the metrics to send.

To enable the Stats Collector Plugin or to configure the type of data sent, use the **dsconfig** command or the Administrative Console. For example, to enable the Stats Collector Plugin to send host CPU metric, memory metrics, and server status metrics using **dsconfig**:

```
dsconfig set-plugin-prop \
    --plugin-name "Stats Collector" \
    --set enabled:true \
    --set host-info:cpu \
    --set host-info:disk \
    --set status-summary-info:basic
```

If you are not using Data Metrics Server to monitor your server, you can disable the generation of some metrics files that are not necessary for the StatsD Monitoring Endpoint. To do this, set the `generate-collector-files` property on the Stats Collector Plugin to `false`.

If you are using the Administrative Console:

1. Click **Show Advanced Configuration**.
2. In the **LDAP (Administration and Monitoring)** section, click **Plugin Root**
3. Edit the **Stats Collector** plugin.

After you enable the Stats Collector and create the StatsD monitoring endpoint, you can:

- Use the data with Splunk as explained in *Sending Metrics to Splunk with StatsD* on page 758.
- Configure other tools that support StatsD, such as CloudWatch or a Prometheus StatsD exporter, to use the data. For more information about this configuration, see your tool's StatsD documentation. Configure the StatsD monitoring endpoint to use the correct host and port. The `dsconfig create-`

`monitoring-endpoint` example above uses a host of localhost and a port of 8125. You can also set these values in the Administrative Console.

Sending Metrics to Splunk with StatsD

About this task

With the StatsD Endpoint type, you can send metric data to a Splunk installation.

In Splunk, you can use SSL to secure ports that are open for StatsD.

> ⓘ **Note:**
>
> StatsD metrics are typically sent over UDP. By using UDP, the client sending metrics does not have to block as it would if using TCP. However, using TCP guarantees order and ensures no metrics are lost.

You can configure open UDP (or TCP) ports in Splunk to accept only connections from a certain hostname or IP address.

To securely send UDP (or TCP) data to Splunk, you can:

Steps

1. Send the data to a Splunk Universal Forwarder.
2. Have the forwarder communicate with the Splunk Indexer over SSL.

Adding custom logged statistics to a Periodic Stats Logger

Add custom statistics based on any attribute in any entry under `cn=monitor` using the Custom Logged Stats object. This configuration object provides powerful controls for how monitor attributes are written to the log. For example, you can extract a value from a monitor attribute using a regular expression. Newly created Custom Logged Stats will automatically be included in the Periodic Stats Logger output.

Besides allowing a straight pass-through of the values using the 'raw' statistic-type, you can configure attributes to be treated as a counter (where the interval includes the difference in the value since the last interval), an average, a minimum, or a maximum value held by the attribute during the specified interval. The value of an attribute can also be scaled by a fixed value or by the value of another monitor attribute.

> ⓘ **Note:** Custom third-party server extensions that were written using the Server SDK can also expose interval statistics using a Periodic Stats Logger. The extension must first implement the SDK's `MonitorProvider` interface and register with the server. The monitor attributes produced by this custom `MonitorProvider` are then available to be referenced by a Custom Logged Stats object.

To illustrate how to configure a Custom Logged Statistics Logger, the following procedure reproduces the built-in "Consumer Total GB" column that shows up in the output when the `included-resource-stat` property is set to memory-utilization on the Periodic Stats Logger. The column is derived from the `total-bytes-used-by-memory-consumers` attribute of the `cn=JVM Memory Usage,cn=monitor` entry as follows:

```
dn: cn=JVM Memory Usage,cn=monitor
objectClass: top
objectClass: ds-monitor-entry
objectClass: ds-memory-usage-monitor-entry
objectClass: extensibleObject
cn: JVM Memory Usage
...
total-bytes-used-by-memory-consumers: 3250017037
```

Configuring a custom logged statistic using dsconfig interactive

Steps

1. Run **dsconfig** and enter the LDAP/LDAPS connection parameters when prompted.

```
$ bin/dsconfig
```

2. On the Directory Server configuration main menu (Advanced Objects menu), enter the number corresponding to Custom Logged Stats.
3. On the Custom Logged Stats menu, enter the number corresponding to Create a new Custom Logged Stats.
4. Select the Stats Logger Plugin from the list if more than one is present on the system. If you only have one stats logger, press **Enter** to confirm that you want to use the existing plugin.
5. Enter a descriptive name for the Custom Logged Stats. For this example, enter `Memory Usage`.
6. From the `monitor-objectclass` property menu, enter the objectclass attribute to monitor. For this example, enter `ds-memory-usage-monitor-entry`. You can run **ldapsearch** using the base DN "cn=JVM Memory Usage,cn=monitor" entry to view the entry.
7. Next, specify the attributes of the monitor entry that you want to log in the stats logger. In this example, enter `total-bytes-used-by-memory-consumers`, and then press **Enter** again to continue.
8. Next, specify the type of statistics for the monitored attribute that will appear in the log file. In this example, enter the option for raw statistics as recorded by the logger.
9. In the Custom Logged Stats menu, review the configuration. At this point, we want to set up a column name that lists the Memory Usage. Enter the option to change the `column-name` property.
10. Next, we want to add a specific label for the column name. Enter the option to add a value, and then enter **Memory Consumer Total (GB)**, and press **Enter** again to continue.
11. Confirm that you want to use the `column-name` value that you entered in the previous step, and then press `Enter` to use the value.
12. Next, we want to scale the Memory Consumer Totals by one gigabyte. On the **Custom Logged Stats** menu, enter the option to change the `divide-value-by` property.
13. On the `divide-value-by` property menu, enter the option to change the value, and then enter 1073741824 (i.e., 1073741824 bytes = 1 gigabytes).
14. On the **Custom Logged Stats** menu, review your configuration. When finished, enter `f` to save and apply the settings.

```
>>>> Configure the properties of the Custom Logged Stats
 >>>> via creating 'Memory Usage' Custom Logged Stats

         Property                    Value(s)
         ----------------------------------------------------------------
    1)   description                 -
    2)   enabled                     true
    3)   monitor-objectclass         ds-memory-usage-monitor-entry
    4)   include-filter              -
    5)   attribute-to-log            total-bytes-used-by-memory-consumers
    6)   column-name                 Memory Consumer Total (GB)
    7)   statistic-type              raw
    8)   header-prefix               -
    9)   header-prefix-attribute     -
    10)  regex-pattern               -
    11)  regex-replacement           -
    12)  divide-value-by             1073741824
    13)  divide-value-by-attribute   -
    14)  decimal-format              #.##
    15)  non-zero-implies-not-idle   false

    ?)   help
```

```
    f)   finish - create the new Custom Logged Stats
    a)   hide advanced properties of the Custom Logged Stats
    d)   display the equivalent dsconfig arguments to create this object
    b)   back
    q)   quit

Enter choice [b]:
```

```
The Custom Logged Stats was created successfully
```

When the Custom Logged Stats configuration change is completed, the new stats value should immediately show up in the Stats Logger output file.

Configuring a custom stats logger using dsconfig non-interactive

Steps

- Use the **`dsconfig`** non-interactive command-line equivalent to create your custom stats logger. The following one-line command replicates the procedure in the previous section. This command produces a column named "Memory Consumer Total (GB)" that contains the value of the of `total-bytes-used-by-memory-consumers` attribute pulled from the entry with the `ds-memory-usage-monitor-entry` objectclass. This value is scaled by 1073741824 to get to a value represented in GBs.

```
$ bin/dsconfig create-custom-logged-stats --plugin-name "Stats Logger" \
  --stats-name "Memory Usage" --type custom \
  --set monitor-objectclass:ds-memory-usage-monitor-entry \
  --set attribute-to-log:total-bytes-used-by-memory-consumers \
  --set "column-name:Memory Consumer Total (GB)" --set statistic-type:raw \
  --set divide-value-by:1073741824
```

Updating the Global Configuration

You also need to create Global Configuration-tracked applications for each app (connection criteria) you intend to track. The `tracked-application` property allows individual applications to be identified in the server by connection criteria. The name of the tracked application is the same as the name you defined for the connection criteria.

For example, the following **`dsconfig`** command line adds the connection criteria we created in the previous step to the list of tracked applications:

```
$ bin/dsconfig set-global-configuration-prop \
  --set "tracked-application:Directory Manager”
```

The value of the `tracked-application` field corresponds to the value of the `application-name` dimension value that clients will specify when accessing the data via the API.

Managing Splunk

You can use Splunk to monitor Directory Server metrics.

You can monitor Directory Server with StatsD or the Splunk Universal Forwarder.

Monitoring Directory Server metrics with Splunk

Metrics from Directory Server can be sent to Splunk through either StatsD or a Periodic Stats Logger

Ping provides a free Splunk (*https://splunkbase.splunk.com/app/5523/*) application for Directory Server that customers can use to create dashboards.

Metrics from Directory Server can be sent to Splunk either through StatsD pushed directly to Splunk, or through a Periodic Stats Logger that is monitored by a Splunk Universal Forwarder.

Sending Metrics with StatsD
Monitoring Directory Server using StatsD and Splunk.

In order to send metrics from Directory Server to Splunk with StatsD, you will need to create a StatsD Monitoring Endpoint. On the Splunk side you will need to open the corresponding port to receive the metrics.

Configuring a StatsD monitoring endpoint
The Stats Collector is used to create a StatsD mnitoring endpoint.

About this task

Steps

1. Run the following command to enable the Stats Collector plugin.

```
dsconfig set-plugin-prop --plugin-name "Stats Collector" --set
 enabled:true
```

2. Run the following command to create a StatsD monitoring endpoint. After running the following command, Directory Server will send StatsD messages to your Splunk server over UDP at port 8125.

```
dsconfig create-monitoring-endpoint --endpoint-name StatsDEndpoint --type
 statsd \
    --set enabled:true --set hostname:your-splunk.example.com \
--set server-port:8125 --set connection-type:unencrypted-udp
```

Configuring Splunk to receive StatsD metrics
Splunk can be used to receive StatsD metrics.

About this task

Run the following commands to configure Splunk to receive StatsD metrics.

Steps

1. Log in to your Splunk dashboard at `http://your-splunk.example.com:8000`.
2. Go to **Settings** > **Data Inputs**.
3. Under Local inputs, click **UDP**.
4. Click **New Local UDP**.
5. Enter the following information:
   Port:`8125`
6. Click **Next**.
7. Set Source type to **Metrics** > **statsd**.
8. Create a new metrics index for the input. Make a note of the index name you choose.
9. Set the App Context to `PingDirectory App for Splunk`.
10. Click **Review**.
11. Slick **Submit**.

ⓘ **Note:** StatsD metrics are typically sent over UDP. Using UDP, the client sending metrics does not have to block as it would if using TCP. However, using TCP guarantees order and ensures no metrics

are lost. To more securely send metrics to Splunk, you can first send your StatsD metrics to a Splunk Universal Forwarder, and have the forwarder communicate with Splunk over SSL

Sending Metrics with the Periodic Stats Logger and the Splunk Universal Forwarder
A Splunk Universal Forwarder can be used to forward Directory Server events to Splunk.

The Periodic Stats Logger plugin writes Directory Server metrics to a file in the server root. A Splunk Universal Forwarder can be configured to watch this file and forward the metrics to Splunk.

Configuring the Periodic Stats Logger
Directory Server can log various metrics, including LDAP operation counts and GC activity, to a JSON-formatted log.

About this task

To configure Directory Server to log to a JSON-formatted log file with various metrics, including LDAP operation counts and GC activity, run the following command. The metrics will be written to `logs/dsstats.json` in the server root.

```
dsconfig set-plugin-prop \
    --plugin-name "Stats Logger" \
    --set enabled:true \
    --set log-file:logs/dsstats.json \
    --set log-file-format:json \
    --set local-db-backend-info:extended \
    --set header-prefix-per-column:true \
    --set empty-instead-of-zero:false \
    --set per-application-ldap-stats:per-application-and-aggregate
```

Configuring the Splunk Universal Forwarder
You can use the Splunk Universal Forwarder to monitor the `logs/dsstats.json` file and forward the metrics to Splunk.

About this task

The Splunk Universal Forwarder can be configured to monitor the `logs/dsstats.json` file and forward the metrics to Splunk. Under the Splunk Forwarder application files, update the `etc/apps/search/local/inputs.conf` file with the following configuration for `dsstats.json`:

```
[monitor:///path/to/PingDirectory/logs/dsstats.json]
sourcetype=log2metrics_json
index=pdmetrics
disabled=false
host=pd1
```

The index name can be customized, and the host value can be configured to label where the metrics are being forwarded from.

Using the Directory Server app for Splunk

You can access several dashboards for monitoring Directory Server.

When you have installed the Directory Server application for Splunk, you will have access to several dashboards for monitoring metrics. To use these dashboards, you will need to select the index name and metrics source you configured. If you are sending metrics with StatsD, use the metrics index you defined when creating the UDP input in Splunk, and select the StatsD radio button at the top of each dashboard. If you are sending metrics with the Periodic Stats Logger, use the index you defined in the Splunk Universal Forwarder configuration and select the **dsstats.json** radio button at the top of each dashboard.

Managing Notifications and Alerts

The PingDirectory Server provides delivery mechanisms for account status notifications and administrative alerts using SMTP, JMX, or SNMP in addition to standard error logging. Alerts and events reflect state changes within the server that may be of interest to a user or monitoring service. Notifications are typically the delivery of an alert or event to a user or monitoring service. Account status notifications are only delivered to the account owner notifying a change in state in the account.

This chapter presents the following topics:

Working with Account Status Notifications

The PingDirectory Server supports notification handlers that can be used to notify users and/or administrators of significant changes related to password policy state for user entries. The following two notification handlers are available:

- **Error Log Account Status Notification Handler**. Enabled by default. The handlers send alerts to the error log when an account event occurs.
- **SMTP Account Status Notification Handler**. Disabled by default. You can enable the SMTP Handler with the **`dsconfig`** command to send notifications to designated email addresses.

Account Status Notification Types

The handlers send alerts when one of the account status events described in the following table occurs during password policy processing.

Account Status Notification Types

Account Status Notification Types	Description
account-disabled	Generates a notification whenever a user account is disabled by an administrator.
account-enabled	Generates a notification whenever a user account is enabled by an administrator.
account-expired	Generates a notification whenever a user authentication attempt fails because the account has expired.
account-idle-locked	Generates a notification whenever a user authentication attempt fails because the account has been locked after idling for too long.
account-permanently-locked	Generates a notification whenever a user account is permanently locked (requiring administrative action to unlock the account) after too many failed attempts.
account-reset-locked	Generates a notification whenever an authentication attempt fails because the user account is locked because the user failed to change a password within the required interval that was reset by an administrator.
account-temporarily-locked	Generates a notification whenever a user account is temporarily locked after too many failed attempts.
account-unlocked	Generates a notification whenever a user account is unlocked by an administrator.
password-changed	Generates a notification whenever a user changes his or her own password.

Account Status Notification Types	Description
password-expired	Generates a notification whenever a user authentication fails because the password has expired.
password-expiring	Generates a notification the first time that a password expiration warning is encountered for a user password.
password-reset	Generates a notification whenever a user's password is reset by an administrator.

Working with the Error Log Account Status Notification Handler

The Error Log Account Status Notification Handler is enabled by default and sends alerts when one of the account status events occur.

To Disable the Error Log Account Status Notification Handler

Steps

- Use the **dsconfig** tool to disable the Error Log Handler. You can view the log at `logs/error`.

```
$ bin/dsconfig set-account-status-notification-handler-prop \
  ---handler-name "Error Log Handler" --set enabled:false
```

To Remove a Notification Type from the Error Log Handler

Steps

- While not recommended, if you want to remove an account status notification type, use the **dsconfig** tool with the `--remove` option.

```
$ bin/dsconfig set-account-status-notification-handler-prop \
  --handler-name "Error Log Handler" \
  --remove account-status-notification-type: password-reset
```

Working with the SMTP Account Status Notification Handler

You can enable account status notifications to be sent to designated email addresses of end users, administrators, or both through an outgoing SMTP server. The email message is automatically generated from template files that contain the text to use in the message body. For example, the message subject for the account-disabled event is:

```
account-disabled: Your directory account has been disabled.
```

The message templates are located in the `config/messages` directory. The typical message body template is as follows:

```
Your directory account has been disabled.

For further assistance, please contact a server administrator.
```

By default, the sender address is notifications@example.com, but you can configure your own address.

Before you enable the SMTP Account Status Notification Handler, you must configure the Directory Server to use at least one mail server as shown below. You can configure an SMTP server using **dsconfig** and the `set-global-configuration-prop` option.

To Configure the SMTP Server

Steps

1. Use **dsconfig** to configure a simple mail server.

```
$ bin/dsconfig create-external-server --server-name smtp1 \
  --type smtp --set server-host-name:smtp.example.com
```

2. Use **dsconfig** to configure an SMTP server. This command adds the server to the global list of mail servers that the Directory Server can use.

```
$ bin/dsconfig set-global-configuration-prop \
  --set smtp-server:smtp1
```

To Configure a StartTLS Connection to the SMTP Server

Steps

1. Use **dsconfig** to configure a StartTLS connection to the server.

```
$ bin/dsconfig create-external-server \
  --server-name myTLSServer --type smtp \
  --set server-host-name:tls.smtp.example.com \
  --set server-port:587 \
  --set smtp-security:starttls \
  --set user-name:MyAccountName \
  --set password:AAD5yZ+DjvwiYkBSMer6GQ6B3szQ6gSSBjA=
```

2. Use **dsconfig** to configure a newly-created SMTP server. This command adds the server to the global list of mail servers that the Directory Server can use.

```
$ bin/dsconfig set-global-configuration-prop \
  -set smtp-server:myTLSServer
```

To Configure an SSL Connection to the SMTP Server

Steps

1. Use **dsconfig** to create an external SMTP server using SSL.

```
$ bin/dsconfig create-external-server \
  --server-name ssl.smtp.example.com \
  --type smtp --set server-host-name:stmp.gmail.com \
  --set server-port:465 \
  --set smtp-security:ssl \
  --set "user-name:my.name@example.com" \
  --set password:xxxxxx --set "smtp-timeout:10s"
```

2. Use **dsconfig** to configure an SMTP server. This command adds the server to the global list of mail servers that the Directory Server can use.

```
$ bin/dsconfig set-global-configuration-prop \
  --set smtp-server:ssl.smtp.example.com
```

To Enable the SMTP Account Status Notification Handler

Steps

- Use **dsconfig** to enable the SMTP account status notification handler.

```
$ bin/dsconfig set-account-status-notification-handler-prop \
  --handler-name "SMTP Handler" --set enabled:true \
  --set "recipient-address:admin@example.com" \
```

```
  --set "sender-address:acct-status-notifications@example.com
```

To View the Account Status Notification Handlers

Steps

- After you have enabled the SMTP server, view the list of account status notification handlers using **`dsconfig`**.

```
$ bin/dsconfig list-account-status-notification-handlers
```

```
Account Status Notification Handler : Type   : enabled
------------------------------------:-----------:--------
Error Log Handler                   : error-log : true
SMTP Handler                        : smtp      : true
```

Associating Account Status Notification Handlers with Password Policies

To generate notifications whenever appropriate password policy state changes occur in the server, the password policy that governs the entry being updated must be configured to use one or more account status notification handlers. By default, password policies are not configured with any such handlers, and therefore, no account status notifications will be generated.

The set of account status notification handlers that should be in use for a password policy is controlled by the `account-status-notification-handler` property for that password policy. It can be configured using **`dsconfig`** or the Administrative Console. For example, the following change updates the default password policy, so that the error log account status notification handler will be invoked for any appropriate password policy state changes for entries governed by the default password policy:

```
$ bin/dsconfig set-password-policy-prop \
  --policy-name "Default Password Policy" \
  --set "account-status-notification-handler:Error Log Handler"
```

Working with Administrative Alert Handlers

The PingDirectoryProxy Server provides mechanisms to send alert notifications to administrators when significant problems or events occur during processing, such as problems during server startup or shutdown. The Directory Proxy Server provides a number of alert handler implementations, including:

- **Error Log Alert Handler**. Sends administrative alerts to the configured server error logger(s).
- **Exec Alert Handler**. Executes a specified command on the local system if an administrative alert matching the criteria for this alert handler is generated by the Directory Proxy Server. Information about the administrative alert will be made available to the executed application as arguments provided by the command.
- **Groovy Scripted Alert Handler**. Provides alert handler implementations defined in a dynamically-loaded Groovy script that implements the `ScriptedAlertHandler` class defined in the Server SDK.
- **JMX Alert Handler**. Sends administrative alerts to clients using the Java Management Extensions (JMX) protocol. Ping Identity uses JMX for monitoring entries and requires that the JMX connection handler be enabled.
- **SMTP Alert Handler**. Sends administrative alerts to clients via email using the Simple Mail Transfer Protocol (SMTP). The server requires that one or more SMTP servers be defined in the global configuration.
- **SNMP Alert Handler**. Sends administrative alerts to clients using the Simple Network Monitoring Protocol (SNMP). The server must have an SNMP agent capable of communicating via SNMP 2c.
- **SNMP Subagent Alert Handler**. Sends SNMP traps to a master agent in response to administrative alerts generated within the server.
- **Third Party Alert Handler**. Provides alert handler implementations created in third-party code using the Server SDK.

Administrative Alert Types

If enabled, the Directory Server can generate administrative alerts when the events occur. A full listing of system alerts and their severity is available in `<server-root>/docs/admin-alerts-list.csv`

Configuring the JMX Connection Handler and Alert Handler

You can configure the JMX connection handler and alert handler respectively using the **`dsconfig`** tool. Any user allowed to receive JMX notifications must have the `jmx-read` and `jmx-notify` privileges. By default, these privileges are not granted to any users (including root users or global administrators). For security reasons, we recommend that you create a separate user account that does not have any other privileges but these. Although not shown in this section, you can configure the JMX connection handler and alert handler using **`dsconfig`** in interactive command-line mode, which is visible on the "Standard" object menu.

Configuring the JMX Connection Handler

Steps

1. Use **`dsconfig`** to enable the JMX Connection Handler.

```
$ bin/dsconfig set-connection-handler-prop \
  --handler-name "JMX Connection Handler" \
  --set enabled:true \
  --set listen-port:1689
```

2. Add a new non-root user account with the `jmx-read` and `jmx-notify` privileges. This account can be added using the **`ldapmodify`** tool using an LDIF representation like:

```
dn: cn=JMX User,cn=Root DNs,cn=config
changetype: add
objectClass: top
objectClass: person
objectClass: organizationalPerson
objectClass: inetOrgPerson
objectClass: ds-cfg-root-dn-user
givenName: JMX
sn: User
cn: JMX User
userPassword: password
ds-cfg-inherit-default-root-privileges: false
ds-cfg-alternate-bind-dn: cn=JMX User
ds-privilege-name: jmx-read
ds-privilege-name: jmx-notify
```

Configuring the JMX Alert Handler

Steps

- Use **`dsconfig`** to configure the JMX Alert Handler.

```
$ bin/dsconfig set-alert-handler-prop --handler-name "JMX Alert Handler" \
  --set enabled:true
```

Configuring the SMTP Alert Handler

By default, there is no configuration entry for an SMTP alert handler. To create a new instance of an SMTP alert handler, use the **`dsconfig`** tool.

Configuring the SMTP Alert Handler

Steps

- Use the **dsconfig** tool to configure the SMTP Alert Handler.

```
$ bin/dsconfig create-alert-handler \
  --handler-name "SMTP Alert Handler" \
  --type smtp \
  --set enabled:true \
  --set "sender-address:alerts@example.com" \
  --set "recipient-address:administrators@example.com" \
  --set "message-subject:Directory Admin Alert \%\%alert-type\%\%" \
  --set "message-body:Administrative alert:\\n\%\%alert-message\%\%"
```

Configuring the SNMP Subagent Alert Handler

You can configure the SNMP Subagent alert handler using the **dsconfig** tool, which is visible at the "Standard" object menu. Before you begin, you need an SNMP Subagent capable of communicating via SNMP2c. For more information on SNMP, see Monitoring Using SNMP.

To Configure the SNMP Subagent Alert Handler

Steps

- Use **dsconfig** to configure the SNMP subagent alert handler. The `server-host-name` is the address of the system running the SNMP subagent. The `server-port` is the port number on which the subagent is running. The `community-name` is the name of the SNMP community that is used for the traps.

  The Directory Server also supports a SNMP Alert Handler, which is used in deployments that do not enable an SNMP subagent.

```
$ bin/dsconfig set-alert-handler-prop \
  --handler-name "SNMP Subagent Alert Handler" \
  --set enabled:true \
  --set server-host-name:host2 \
  --set server-port:162 \
  --set community-name:public
```

E-mail Account Status Notification Handler

The e-mail account status notification handler enables the sending of messages that can contain either a plain-text body, an HTML-formatted body, or both (in which case the recipient's email client determines which version to display). Messages can optionally include a set of attachments.

The `config/account-status-notification-email-templates` directory contains a set of template files that can be used to generate multi-part email messages to send to end users and server administrators when certain events occur that affect the state of a user account. A separate template is used for each type of event so that the message contents, recipients, and other details can be customized for specific events.

Account Status Notification Types

The following account status notification types are available.

- account-temporarily-locked -- A user's account has been temporarily locked due to too many failed authentication attempts. The account will remain locked until a configured length of time elapses, or until the password is reset by an administrator.
- account-permanently-locked -- A user's account has been permanently locked due to too many failed authentication attempts. The account will remain locked until the password is reset by an administrator.

- account-idle-locked -- An authentication attempt failed because it has been too long (longer than the `idle-lockout-interval` configured in the associated password policy) since the user last successfully authenticated to the server. The account will remain locked until the password is reset by an administrator.
- account-reset-locked -- An authentication attempt failed because the user's account was in a "must change password" state following an administrative password reset, but the user did not choose a new password in a timely manner (within the `max-password-reset-age` duration configured in the associated password policy). The account will remain locked until the password is reset again by an administrator.
- account-unlocked -- A locked user account has been unlocked (for example, by an administrative password reset).
- account-disabled -- A user's account has been administratively disabled (by setting the `ds-pwp-account-disabled` operational attribute to `true` in the user entry). The user will not be allowed to authenticate until this attribute is removed or its value is set to `false`.
- account-enabled -- A user's account has been administratively enabled (by setting the `ds-pwp-account-disabled` operational attribute to `false` in the user entry, or by removing this attribute from the entry).
- account-not-yet-active -- An authentication attempt failed because the user account is configured with an activation time (via the `ds-pwp-account-activation-time` operational attribute in the user's entry) that is in the future. The user will not be allowed to authenticate until this time arrives, until the activation time is removed, or until the activation time is set to a time in the past.
- account-expired -- An authentication attempt failed because the user account is configured with an expiration time (via the `ds-pwp-account-expiration-time` operational attribute in the user's entry) that is in the past. The user will not be allowed to authenticate until the expiration time is removed or set to a time in the future.
- password-expired -- An authentication attempt failed because the user's password has expired. The user will not be allowed to authenticate until their password is reset by an administrator (or until they change their own password if `allow-expired-password-changes` is set to `true` in the associated password policy).
- password-expiring -- The user successfully authenticated, but their password will expire in the near future (as determined by the `password-expiration-warning-interval` setting in the associated password policy). This notification type will only be generated the first time that a user authenticated within a given warning interval.
- password-reset -- A user's password has been reset by an administrator.
- password-changed -- A user changed their own password.
- account-created -- A new account was created in an add request that matches the criteria specified in the `account-creation-notification-request-criteria` property of the account status notification handler configuration.
- account-updated -- An existing account was updated in a modify or modify DN request that matches the criteria specified in the `account-update-notification-request-criteria` property of the account status notification handler configuration.

Message Template File Format

A message template file can contain the following sections:

- A required header section describing the sender, recipients, subject, and other header elements. This section must appear exactly once, and it must be the first section of the template file.
- An optional section providing a plain-text version of the message body. This section may appear at most once, and at least one of the plain-text and HTML-formatted body sections must be present.
- An optional section providing an HTML-formatted version of the message body. This section may appear at most once, and at least one of the plain-text and HTML-formatted body sections must be present.

- An optional section providing information about an attachment to include in the message. This section can be included zero, one, or multiple times, and all attachment sections must appear after all other sections.

A message template file must start with a header section that specifies the recipient and sender addresses, the message subject, and the values of any custom headers that should appear in the message. It also indicates whether the template should be enabled and used to send email messages.

The header section must begin with the line `----- BEGIN HEADERS -----`, and may contain a number of name-value pairs that provide information about the message. Each name-value pair should appear on a separate line, and the name and value must be separated by a colon and optional spaces. The header section can include the following:

- ENABLED -- Specifies whether this template is enabled and should be used to generate an email message. This element must be present and must have a value of either `true` or `false`. This is primarily intended for cases in which a message should only be generated under certain circumstances. If you do not ever want a message sent for a particular notification type, you should not specify a template file for that notification type in the configuration for the multi-part email account status notification handler.
- TO -- Specifies a primary recipient for the email message. This recipient will be visible to all recipients. The value can be either an email address (for example, `jdoe@example.com`) or a name followed by the email address in angle brackets (for example, `John Doe <jdoe@example.com>`). To specify multiple TO recipients, this element can be specified multiple times.. It can be omitted if the message should have only CC or BCC recipients, but the message must have at least one TO, CC, or BCC recipient.
- CC -- Specifies a carbon-copied recipient for the email message. This recipient will be visible to all recipients. The value can be either an email address (for example, `jdoe@example.com`) or a name followed by the email address in angle brackets (for example, `John Doe <jdoe@example.com>`). This element can be specified multiple times to specify multiple `CC` recipients. It can be omitted if the message should have only TO or BCC recipients, but the message must have at least one TO, CC, or BCC recipient.
- BCC -- Specifies a blind carbon-copied recipient for the email message. This recipient will not be visible to other recipients. The value can be either an email address (for example, `jdoe@example.com`) or a name followed by the email address in angle brackets (for example, `John Doe <jdoe@example.com>`). This element can be specified multiple times to specify multiple BCC recipients. It can be omitted if the message should have only TO or CC recipients, but the message must have at least one TO, CC, or BCC recipient.
- FROM -- Specifies the address from which the email message should appear to be sent. This value can be either an email address (for example, `jdoe@example.com`) or a name followed by the email address in angle brackets (for example, `John Doe <jdoe@example.com>`). This element must be specified exactly once.
- REPLY-TO -- Specifies the default address to which replies should be sent. The value can be either an email address (for example, `jdoe@example.com`) or a name followed by the email address in angle brackets (for example, `John Doe <jdoe@example.com>`). This element is optional and can only be specified once.
- SUBJECT -- Specifies the subject for the email message. This element must be specified exactly once.
- HEADER -- Defines a custom header that should be included in the email message. The header value itself should be a name-value pair with the name and value separated by a colon and optional spaces (for example, *X-Custom-Header-Name*`:`*Custom Header Value*). This is an optional element and can be specified multiple times to specify multiple headers with the same or different names.

The header section must be followed by plain-text and HTML-formatted body sections. Each of these sections consists of free-form text that can span multiple lines and continues until the beginning of the next section or the end of the template file is reached.

If a plain-text body is included, it must begin with the line `----- BEGIN PLAIN-TEXT BODY -----`.
If an HTML-formatted body is included, it must begin with the line `----- BEGIN HTML BODY -----`.
At least one of these sections must be included. If an HTML-formatted body section is included, it is

recommended that a plain-text body also be included for the benefit of clients that prefer plain text or cannot handle HTML content. Any blank lines at the beginning or end of the body text will be removed, but blank lines in the middle of the body are preserved.

The template can also contain any number of optional attachment sections that describe any attachments that should be included in the message. Each attachment must be specified in its own section at the end of the template, after the header, plain-text, and HTML-formatted body sections. Each attachment section must begin with the line `----- BEGIN ATTACHMENT -----` and can include the following elements (using the same colon-delimited syntax used in the header section):

- FILENAME -- The name (without any path information) of the file to be attached. This must be specified, and the file must exist in the same directory as the template file.
- CONTENT-TYPE -- The MIME type for the attachment. This is an optional element. If it is not specified, a default content-type of `application/octet-stream` is used.
- INLINE -- If `true` is specified, indicates that the attachment should be an inline attachment (for example, an image that should be referenced within the HTML body, in which case the image source path should be specified as `cid:` followed by the filename (for example, `cid:*company-logo.png*`). If present, this value must be either `true` or `false`. If it is omitted, a default value of `false` is assumed.

Customizing the Message Content

Before the server parses the template file to generate a message, it pre-processes the file with the Apache Velocity templating engine (*https://velocity.apache.org/*). This enables you to customize the message content with information from the account status notification, details from the associated user entry, and other information. The following variables are defined for use with custom directives:

- $account_entry -- The entry for the user associated with the account status notification.
- $notification_type -- The name of the notification type for the account status notification.
- $notification_message -- A human-readable message that provides a basic description of the account status notification.
- $before_entry -- A version of the associated entry as it appeared before the operation was processed. This is only available for notifications generated as a result of modify or modify DN operations.
- $after_entry -- A version of the associated entry as it appeared after the operation was processed. This is only available for notifications generated as a result of modify or modify DN operations.

In addition to the built-in directives provided by Velocity, the following custom directives are also available:

- #getEntryDN(*entry*, *variableName*) -- Retrieves a string representation of the provided entry's DN and stores it in the specified Velocity context variable.
- #getHasAttribute(*entry*, *attributeName*, *variableName*) -- Determines if the provided entry includes the specified attribute. If it contains the specified attribute, a value of `true` is stored in the specified Velocity context variable. If not, a value of `false` is stored.
- #getUserAttributeNames(*entry*, *variableName*) -- Retrieves a list of the names of the user attributes contained in the provided entry and stores this list in the specified Velocity context variable.
- #getOperationalAttributeNames(*entry*, *variableName*) -- Retrieves a list of the names of the operational attributes contained in the provided entry and stores this list in the specified Velocity context variable.
- #getAttributeValue(*entry*, *attributeName*, *variableName*) -- Retrieves the first value for the indicated attribute from the provided entry and stores it in the specified Velocity context variable. If the attribute is not present in the entry, the variable is removed from the context.
- #getAttributeValues(*entry*, *attributeName*, *variableName*) -- Retrieves the list of values for the indicated attribute from the provided entry and stores this list in the specified Velocity context variable. If the attribute is not present in the entry, an empty list is stored.
- #getEntryMatchesLDAPFilter(*entry*, *filterString*, *variableName*) -- Determines if the provided entry matches the LDAP filter with the given string representation. If the entry matches, a value of `true` is stored in the Velocity context variable. Otherwise, a value of `false` is stored.
- #getJSONObjectValue(*entry*, *attributeName*, *variableName*) -- Retrieves the first value that can be parsed as a JSON object for the indicated attribute and stores this object in the specified variable in

the Velocity context. If the attribute does not exist in the provided entry, or if none of its values can be parsed as a JSON object, the variable is removed from the context.

- #getJSONObjectValues(*entry*, *attributeName*, *variableName*) -- Retrieves a list of all values that can be parsed as JSON objects for the indicated attribute and stores this list in the specified variable in the Velocity context. If the attribute does not exist in the provided entry, or if none of its values can be parsed as JSON objects, an empty list is stored.
- #getJSONFieldValue(*jsonObject*, *fieldName*, *variableName*) -- Retrieves the first value for the indicated field from the provided JSON object and stores its string representation in the specified Velocity context variable. Nested fields can be targeted by using the period as a delimiter between field names (for example, `name.first` targets the `first` field inside a JSON object that is a value for the outer `name` field). If the field does not exist in the provided object, the variable is removed from the context.
- #getJSONFieldValues(*jsonObject*, *fieldName*, *variableName*) -- Retrieves all values for the indicated field from the provided JSON object and stores a list of their string representations in the specified Velocity context variable. Nested fields can be targeted by using the period as a delimiter between field names (for example, `name.first` targets the `first` field inside a JSON object that is a value for the outer `name` field). If the field does not exist in the provided object (or if its value is an empty array), an empty list is stored.
- #getJSONObjectMatchesFilter(*jsonObject*, *jsonObjectFilterString*, *variableName*) -- Determines if the provided JSON object matches the JSON object filter with the given string representation. If the JSON object matches, a value of `true` is stored in the specified Velocity context variable. Otherwise, a value of `false` is stored.
- #getEntry(*entryDN*, *variableName*) -- Retrieves the entry with the provided DN from the server and stores it in the specified Velocity context variable. If the entry does not exist, the variable is removed from the context.
- #searchForEntries(*baseDN*, *scopeString*, *filterString*, *variableName*) -- Performs an internal search with the provided base DN, scope (which must be one of `base`, `one`, `sub`, or `subordinates`), and filter, and stores the list of matching entries in the specified variable in the context. If the search does not match any entries, if it matches more than 20 entries, or if the search criteria is not indexed, then an empty list is stored.
- #getParentDN(*entryDN*, *variableName*) -- Determines the parent DN for the provided entry DN and stores it in the specified Velocity context variable. If the provided DN contains only a single component, an empty string is stored. If the provided DN is the null DN (and therefore does not have a parent), the variable is removed from the context.
- #getIsAttributeModified(*beforeEntry*, *afterEntry*, *attributeName*, *variableName*) -- Determines if the indicated attribute has been modified between the provided before and after representations of the entry. If the attribute has been modified, a value of `true` is stored in the specified Velocity context variable. Otherwise, a value of `false` is stored.
- #getIsAnyAttributeModified(*beforeEntry*, *afterEntry*, *attributeName1*, *attributeName2*, ..., *variableName*) -- Determines if at least one of the listed attributes has been modified between the provided before and after representations of the entry. If at least one of the listed attributes has been modified, a value of `true` is stored in the specified Velocity context variable. Otherwise, a value of `false` is stored. The first two arguments to this directive must be the before and after representations of the target entry, and the last argument must be the name of the variable in which to store the result. All arguments in between these are treated as the names or OIDs of the attributes for which to make the determination. At least one attribute name or OID must be provided.
- #getModifiedAttributeNames(*beforeEntry*, *afterEntry*, *variableName*) -- Determines which attributes have been modified between the provided before and after representations of the entry and stores a list of their names in the specified Velocity context variable. If the provided entries are identical, an empty list is stored.
- #getModifiedAttributeAddedValues(*beforeEntry*, *afterEntry*, *attributeName*, *variableName*) -- Retrieves a list of the values that have been added between the provided before and after representations of an entry for the indicated attribute and stores this list in the specified Velocity context variable. If the attribute was not altered, or if values were only removed from the attribute, an empty list is stored.
- #getModifiedAttributeRemovedValues(*beforeEntry*, *afterEntry*, *attributeName*, *variableName*) -- Retrieves a list of the values that have been removed between the provided before and after

representations of an entry for the indicated attribute and stores this list in the specified Velocity context variable. If the attribute was not altered, or if values were only added to the attribute, an empty list is stored.

- #getOperationMatchesConnectionCriteria(*criteriaDN*, *variableName*) -- Determines if the operation that triggered the notification matches the connection criteria defined in the entry with the specified DN. If it matches, a value of `true` is stored in the specified Velocity context variable. Otherwise, a value of `false` is stored.
- #getOperationMatchesRequestCriteria(*criteriaDN*, *variableName*) -- Determines if the operation that triggered the notification matches the request criteria defined in the entry with the specified DN. If it matches, a value of `true` is stored in the specified Velocity context variable. Otherwise, a value of `false` is stored.
- #getNotificationPropertyNames(*variableName*) -- Retrieves a list of the names of the account status notification properties that are defined for the notification and stores this list in the specified Velocity context variable.
- #getNotificationPropertyValue(*propertyName*, *variableName*) -- Retrieves the first value for the given account status notification property and stores it in the specified Velocity context variable. If the property is not defined, the variable is removed from the context.
- #getNotificationPropertyValues(*propertyName*, *variableName*) -- Retrieves the list of values for the given account status notification property and stores this list in the specified Velocity context variable. If the property is not defined, an empty list is stored.
- #getFormattedTimestamp(*timestamp*, *formatString*, *variableName*) -- Retrieves a representation of the provided timestamp (which must be in generalized time form) formatted with the given format string (which must be compatible with the java.text.SimpleDateFormat class) and stores it in the specified Velocity context variable. The default time zone of the JVM is used.
- #getFormattedTimestamp(*timestamp*, *formatString*, *timeZone*, *variableName*) -- Retrieves a representation of the provided timestamp (which must be in generalized time form) formatted with the given format string (which must be compatible with the java.text.SimpleDateFormat class) in the indicated time zone (which must be compatible with the java.time.ZoneId class) and stores it in the specified Velocity context variable.
- #getFileExistsInTemplateDirectory(*fileName*, *variableName*) -- Determines if a file with the given name (which must not contain path information) exists in the same directory as the template. If the file exists, a value of `true` is stored in the specified Velocity context variable. Otherwise, a value of `false` is stored.

The following account status notification properties are available for use in conjunction with the #getNotificationPropertyValue and #getNotificationPropertyValues directives:

- notification-type -- The name of the account status notification type.
- notification-time -- A generalized time representation of the time that the notification was generated.
- account-dn -- The DN of the entry for the associated user account.
- password-policy-dn -- The DN of the password policy that governs the associated user account.
- password-changed-time -- A generalized time representation of the time the user's password was last changed.
- seconds-since-password-change -- The number of seconds since the account's password was last changed.
- time-since-password-change -- A human-readable duration indicating the length of time since the user's password was last changed.
- account-is-usable -- A value of `true` if the account is considered usable, or `false` if it is not usable (for example, if the account is locked/expired/not yet active, if the password is expired, if the user must choose a new password, etc.).
- account-usability-errors -- A list of human-readable strings providing information about any errors that currently affect the account's usability.
- account-usability-warnings -- A list of human-readable strings providing information about any warning conditions that have the potential to affect the account's usability in the near future.

- account-usability-notices -- A list of human-readable strings providing information about any notable account state conditions that are not expected to affect the account's usability in the near future.
- account-is-disabled -- The value `true` if the account has been administratively disabled, or `false` if it has not been administratively disabvled.
- account-is-not-yet-active -- The value `true` if the account has an activation time in the future, or `false` if not.
- account-activation-time -- A generalized time representation of the account activation time, if defined.
- seconds-until-account-activation -- The number of seconds until the account becomes active if it has an activation time that is in the future.
- time-until-account-activation -- A human-readable duration indicating the length of time until the account becomes active if it has an activation time that is in the future.
- seconds-since-account-activation -- The number of seconds since the account became active if it has an activation time that is in the past.
- time-since-account-activation -- A human-readable duration indicating the length of time since the account became active if it has an activation time that is in the past.
- account-is-expired -- The value `true` if the account has an expiration time in the past, or `false` if not.
- account-expiration-time -- A generalized time representation of the account expiration time, if defined.
- seconds-until-account-expiration -- The number of seconds until the account expires if it has an expiration time in the future.
- time-until-account-expiration -- A human-readable duration indicating the length of time until the account expires if it has an expiration time in the future.
- seconds-since-account-expiration -- The number of seconds since the account expired if it has an expiration time in the past.
- time-since-account-expiration -- A human-readable duration indicating the length of time since the account expired if it has an expiration time in the past.
- account-is-failure-locked -- The value `true` if the account has been temporarily or permanently locked due to too many failed authentication attempts, or `false` if not.
- failure-locked-time -- A generalized time representation of the time that the account became failure-locked, if applicable.
- account-unlock-time -- A generalized time representation of the time that the account will be automatically unlocked, if it is temporarily locked due to too many failed authentication attempts.
- seconds-until-unlock -- The number of seconds until the account unlock time, if applicable.
- time-until-unlock -- A human-readable duration indicating the length of time until the account unlock time, if applicable.
- failure-lockout-count -- The number of failed authentication attempts required to lock an account, if configured.
- failure-lockout-duration-seconds -- The number of seconds that a failure-locked account will be prevented from authenticating, if defined.
- failure-lockout-duration -- A human-readable duration that indicates how long a failure-locked account will be prevented from authenticating, if defined.
- account-is-idle-locked -- The value `true` if the account is locked because it has been too long since the user last authenticated, or `false` if not.
- idle-lockout-interval-seconds -- The maximum number of seconds that may pass between successful authentications before an account becomes idle-locked, if configured.
- idle-lockout-interval -- A human-readable duration that indicates how much time may pass between successful authentications before an account becomes idle-locked, if configured.
- last-login-time -- A generalized time representation of the time that the user last successfully authenticated, if available.
- seconds-since-last-login -- The number of seconds that have passed since the user last successfully authenticated, if available.
- time-since-last-login -- A human-readable duration indicating the length of time that has passed since the user last successfully authenticated, if available.
- last-login-ip-address -- The IP address of the client from which the user last authenticated, if available.

- account-is-reset-locked -- The value `true` if the account is locked because the user failed to choose a new password in a timely manner after an administrative password reset, or `false` if not.
- maximum-password-reset-age-seconds -- The maximum number of seconds that a user has to choose a new password after an administrative password reset before the account becomes reset-locked, if defined.
- maximum-password-reset-age -- A human-readable duration indicating the maximum length of time that a user has to choose a new password after an administrative password reset before the account becomes reset-locked, if defined.
- must-change-password -- The value `true` if the user will be required to choose a new password before they are allowed to perform any other operations on the server, or `false` if not.
- account-was-unlocked -- The value `true` if the account had been locked but was just unlocked by the associated operation, or `false` if not.
- password-is-expired -- The value `true` if the account has an expired password, or `false` if not.
- password-is-expiring -- The value `true` if the account has a password that will expire in the near future (as determined by the password-expiration-warning-interval property in the associated password policy), or `false` if not.
- maximum-password-age-seconds -- The maximum number of seconds that an account is permitted to keep the same password before it expires, if defined.
- maximum-password-age -- A human-readable duration indicating the maximum length of time that an account is permitted to keep the same password before it expires, if defined.
- maximum-password-age -- A human-readable duration indicating the maximum length of time that an account is permitted to keep the same password before it expires, if defined.
- password-expiration-time -- A generalized time representation of the time that the account's password did or will expire, if applicable.
- seconds-until-password-expiration -- The number of seconds until the account's password will expire, if it has an expiration time that is in the future.
- time-until-password-expiration -- A human-readable duration indicating the length of time until the account's password will expire, if it has an expiration time that is in the future.
- seconds-since-password-expiration -- The number of seconds since the account's password expired, if it has an expiration time that is in the past.
- time-since-password-expiration -- A human-readable duration indicating the length of time since the account's password expired, if it has an expiration time that is in the past.
- old-password -- The clear-text password that the account had before the associated operation was processed, if available.
- new-password -- The clear-text password that the account has after the associated operation was processed, if available.
- new-generated-password -- The clear-text password that was generated for the account by the associated operation, if applicable.
- requester-ip-address -- The IP address of the client that requested the associated operation, if available.
- requester-dn -- The DN of the account that requested the associated operation.
- operation-type -- The name of the operation type for the associated operation.
- operation-id -- A string representation of the operation ID for the associated operation.
- connection-id -- A string representation of the connection ID for the associated operation.
- server-uuid -- A string representation of the unique identifier generated for the server instance that processed the associated operation.
- instance-name -- The instance name for the server instance that processed the associated operation.

Working with the Alerts Backend

The Directory Server stores recently generated admin alerts in an Alerts Backend under the `cn=alerts` branch. The backend makes it possible to obtain admin alert information over LDAP for use with remote

monitoring. The backend's primary job is to process search operations for alerts. It does not support add, modify, or modify DN operations of entries in the `cn=alerts` backend.

The alerts persist on disk in the `config/alerts.ldif` file so that they can survive server restarts. By default, the alerts remain on disk for seven days before being removed. However, administrators can configure the number of days for alert retention using the **dsconfig** tool. The administrative alerts of Warning level or worse that have occurred in the last 48 hours are viewable from the output of the status command-line tool and in the Administrative Console.

To View Information in the Alerts Backend

Steps

- The following uses **ldapsearch** to view the admin alerts.

```
$ bin/ldapsearch --port 1389 --bindDN "cn=Directory Manager" \
  --bindPassword secret --baseDN cn=alerts "(objectclass=ds-admin-alert)"
```

```
dn: ds-alert-id=3d1857a2-e8cf-4e80-ac0e-ba933be59eca,cn=alerts
objectClass: top
objectClass: ds-admin-alert
ds-alert-id: 3d1857a2-e8cf-4e80-ac0e-ba933be59eca
ds-alert-type: server-started
ds-alert-severity: info
ds-alert-type-oid: 1.3.6.1.4.1.32473.2.11.33
ds-alert-time: 20110126041442.622Z
ds-alert-generator: com.unboundid.directory.server.core.directory.server
ds-alert-message: The Directory Server has started successfully
```

To Modify the Alert Retention Time

Steps

1. Use **dsconfig** to change the maximum time information about generated admin alerts is retained in the Alerts backend. After this time, the information gets purged from the Directory Server. The minimum retention time is 0 milliseconds, which immediately purges the alert information.

```
$ bin/dsconfig set-backend-prop --backend-name "alerts" \
  --set "alert-retention-time: 2 weeks"
```

2. View the property using **dsconfig**.

```
$ bin/dsconfig get-backend-prop --backend-name "alerts" \
  --property alert-retention-time
```

```
Property : Value(s)
---------------------:---------
alert-retention-time : 2 w
```

To Configure Duplicate Alert Suppression

Steps

- Use **dsconfig** to configure the maximum number of times an alert is generated within a particular timeframe for the same condition. The `duplicate-alert-time-limit` property specifies the length of time that must pass before duplicate messages are sent over the administrative alert framework. The `duplicate-alert-limit` property specifies the maximum number of duplicate alert messages should be sent over the administrative alert framework in the time limit specified in the `duplicate-alert-time-limit` property.

```
$ bin/dsconfig set-global-configuration-prop \
```

```
  --set duplicate-alert-limit:2 \
  --set "duplicate-alert-time-limit:3 minutes"
```

Working with alarms, alerts, and gauges

An alarm represents a stateful condition of the server or a resource that may indicate a problem, such as low disk space or external server unavailability. A gauge defines a set of threshold values with a specified severity that, when crossed, cause the server to enter or exit an alarm state. Gauges are used for monitoring continuous values like CPU load or free disk space (Numeric Gauge), or an enumerated set of values such as 'server unavailable' or ‘server unavailable’ (Indicator Gauge). Gauges generate alarms, when the gauge’s severity changes due to changes in the monitored value. Like alerts, alarms have severity (NORMAL, WARNING, MINOR, MAJOR, CRITICAL), name, and message. Alarms will always have a Condition property, and may have a Specific Problem or Resource property. If surfaced through SNMP, a Probable Cause property and Alarm Type property are also listed. Alarms can be configured to generate alerts when the alarm's severity changes. The Alarm Manager, which governs the actions performed when an alarm state is entered, is configurable through the **dsconfig** tool and Administrative Console. A complete listing of system alerts, alarms, and their severity is available in `<server-root>/docs/admin-alerts-list.csv`.

There are two alert types supported by the server - standard and alarm-specific. The server constantly monitors for conditions that may need attention by administrators, such as low disk space. For this condition, the standard alert is **low-disk-space-warning**, and the alarm-specific alert is **alarm-warning**. The server can be configured to generate alarm-specific alerts instead of, or in addition to, standard alerts. By default, standard alerts are generated for conditions internally monitored by the server. However, gauges can only generate alarm-alerts.

The Directory Server installs a set of gauges that are specific to the product and that can be cloned or configured through the **dsconfig** tool. Existing gauges can be tailored to fit each environment by adjusting the update interval and threshold values. Configuration of system gauges determines the criteria by which alarms are triggered. The Stats Logger can be used to view historical information about the value and severity of all system gauges.

The Directory Server is compliant with the International Telecommunication Union CCITT Recommendation X.733 (1992) standard for generating and clearing alarms. If configured, entering or exiting an alarm state can result in one or more alerts. An alarm state is exited when the condition no longer applies. An `alarm_cleared` alert type is generated by the system when an alarm's severity changes from a non-normal severity to any other severity. An `alarm_cleared` alert will correlate to a previous alarm when the Condition and Resource properties are the same. The Condition corresponds to the Summary column in the `admin-alerts-list.csv` file.

Like the Alerts Backend, which stores information in **cn=alerts**, the Alarm Backend stores information within the **cn=alarms** backend. Unlike alerts, alarm thresholds have a state over time that can change in severity and be cleared when a monitored value returns to normal. Alarms can be viewed with the **status** tool. As with other alert types, alert handlers can be configured to manage the alerts generated by alarms.

To View Information in the Alarms Backend

Steps

- The following uses **ldapsearch** to view alarms. The following displays the listing for the CPU usage alarm.

```
$ bin/ldapsearch --port 1389 --bindDN "cn=Directory Manager" \
  --bindPassword secret --baseDN cn=alarms "(objectclass=ds-admin-alarm)"
```

```
dn: ds-alarm-id=CPU Usage (Percent)-Host System,cn=alarms
dn: ds-alarm-id=CPU Usage (Percent)-Host System,cn=alarms
objectClass: top
objectClass: ds-admin-alarm
ds-alarm-id: CPU Usage (Percent)-Host System
```

```
ds-alarm-condition: CPU Usage (Percent)
ds-alarm-specific-resource: Host System
ds-alarm-severity: CRITICAL
ds-alarm-previous-severity: CRITICAL
ds-alarm-details: Gauge CPU Usage (Percent) for Host System
       has value 99, having had a value of 83.13 in the
       previous interval. The severity is critical, having
       assumed this severity Thu Sep 25 10:24:20 CDT 2014
       when the value crossed threshold 80
ds-alarm-additional-text: If CPU use is high, check the server's current
 workload
       and other processes on this system and make any needed adjustments.
 Reducing
       the load on the system will lead to better response times
ds-alarm-start-time: 20140925152420.004Z
ds-alarm-critical-last-time: 20140925152420.004Z
ds-alarm-critical-total-duration-millis: 0
```

Testing Alerts and Alarms

After alarms and alert handlers are configured, verify that the server takes the appropriate action when an alarm state changes by manually increasing the severity of a gauge. Alarms and alerts can be verified with the **`status`** tool.

Testing alarms and alerts

Steps

1. Configure a gauge with **`dsconfig`** and set the **`override-severity`** property to critical. The following example uses the CPU Usage (Percent) gauge.

```
$ dsconfig set-gauge-prop \
  --gauge-name "CPU Usage (Percent)" \
  --set override-severity:critical
```

2. Run the **`status`** tool to verify that an alarm was generated with corresponding alerts. The **`status`** tool provides a summary of the server's current state with key metrics and a list of recent alerts and alarms. The sample output has been shortened to show just the alarms and alerts information.

```
$ bin/status

                        --- Administrative Alerts ---
 Severity : Time            : Message
 ---------:-----------------:-------------------------------------------------
 Info     : 11/Aug/2014     : A configuration change has been made in the
          : 15:48:46 -0500  : Directory Server:
          :                 : [11/Aug/2014:15:48:46.054 -0500]
          :                 : conn=17 op=73 dn='cn=Directory Manager,cn=Root
          :                 : DNs,cn=config' authtype=[Simple]
 from=127.0.0.1
          :                 : to=127.0.0.1 command='dsconfig set-gauge-prop
          :                 :  --gauge-name 'Cleaner Backlog (Number Of
 Files)'
          :                 : --set warning-value:-1'
 Info     : 11/Aug/2014     : A configuration change has been made in the
          :  15:47:32 -0500 : Directory Server: [11/Aug/2014:15:47:32.547
 -0500]
          :                 : conn=4 op=196 dn='cn=Directory Manager,cn=Root
          :                 : DNs,cn=config' authtype=[Simple]
 from=127.0.0.1
          :                 : to=127.0.0.1 command='dsconfig set-gauge-prop
          :                 : --gauge-name 'Cleaner Backlog (Number Of
 Files)'
```

```
         :                  :  --set warning-value:0'
Error    : 11/Aug/2014      : Alarm [CPU Usage (Percent). Gauge CPU Usage
(Percent)
         :  15:41:00 -0500 : for Host System has
         :                  : a current value of '18.583333333333332'.
         :                  : The severity is currently OVERRIDDEN in the
         :                  : Gauge's configuration to 'CRITICAL'.
         :                  : The actual severity is: The severity is
         :                  : currently 'NORMAL', having assumed this
severity
         :                  : Mon Aug 11 15:41:00 CDT 2014. If CPU use is
high,
         :                  : check the server's current workload and make
any
         :                  : needed adjustments. Reducing the load on the
system
         :                  : will lead to better response times.
         :                  : Resource='Host System']
         :                  : raised with critical severity
Shown are alerts of severity [Info,Warning,Error,Fatal] from the past 48
hours
Use the --maxAlerts and/or --alertSeverity options to filter this list
```

```
                         --- Alarms ---
Severity : Severity Start : Condition : Resource    : Details
         : Time           :           :              :
---------:----------------:-----------:--------------:-----------------------------
Critical : 11/Aug/2014    : CPU Usage : Host System : Gauge CPU Usage
(Percent) for
         : 15:41:00 -0500 : (Percent) :              : Host System
         :                :           :              : has a current value
of
         :                :           :              : '18.785714285714285'.
         :                :           :              : The severity is
currently
         :                :           :              : 'CRITICAL', having
assumed
         :                :           :              : this severity Mon Aug
11
         :                :           :              : 15:49:00 CDT 2014. If
CPU use
         :                :           :              : is high, check the
server's
         :                :           :              : current workload and
make any
         :                :           :              : needed adjustments.
Reducing
         :                :           :              : the load on the
system will
         :                :           :              : lead to better
response times
Warning  : 11/Aug/2014    : Work Queue: Work Queue   : Gauge Work Queue Size
(Number
         : 15:39:40 -0500 : Size      :              : of Requests) for Work
Queue
         :                : (Number of:              : has a current value
of '27'.
         :                : Requests) :              : The severity is
currently
         :                :           :              : 'WARNING' having
assumed this
         :                :           :              : severity Mon Aug 11
15:48:50
```

```
          :                 :           :              : CDT 2014. If all
 worker
          :                 :           :              : threads are busy
 processing
          :                 :           :              : other client
 requests, then
          :                 :           :              : new requests that
 arrive will
          :                 :           :              : be forced to wait in
 the work
          :                 :           :              : queue until a worker
 thread
          :                 :           :              : becomes available
Shown are alarms of severity [Warning,Minor,Major,Critical]
Use the --alarmSeverity option to filter this list
```

Indeterminate alarms

Indeterminate alarms are raised for a server condition for which a severity cannot be determined. In most cases these alarms are benign and do not issue alerts nor appear in the output of the **`status`** tool or Administrative Console by default. These alarms are usually caused by an enabled gauge that is intended to measure an aspect of the server that is not currently enabled. For example, gauges intended to monitor metrics related to replication may produce indeterminate alarms if a Directory Server is not currently replicating data. The gauge can be disabled if needed.

For more information about indeterminate alarms, view the gauge's associated monitor entry. There may be messages that can help determine the issue. The following is sample output from the **`status`** tool run with the **`—alarmSeverity=indeterminate`** option:

```
                        --- Alarms ---
Severity     : Severity Start : Condition      : Resource   : Details
             : Time           :                :            :
-------------:----------------:----------------:------------:-----------------------
Normal       : 26/Aug/2014    : Startup Begun  : cn=config  : The Directory
 Server
             : 14:16:29 -0500 :                :            : is starting.
             :                :                :            :
Indeterminate: 26/Aug/2014    : Replication    : not        : The value of
 gauge
             : 14:16:40 -0500 : Latency        : available  : Replication
 Latency
             :                : (Milliseconds) :            : (Milliseconds)
 could not
             :                :                :            : be determined.
 The
             :                :                :            : severity is
 INDETERMINATE,
             :                :                :            : having assumed
 this
             :                :                :            : severity Tue Aug
 26
             :                :                :            : 14:17:10 CDT
 2014.
```

The following is an indeterminate alarm for the Replication Latency (Milliseconds) gauge. The following is a sample search of the monitor backend for this gauge’s entry. The result is an error message may explain the indeterminate severity:

```
# ldapsearch -w password --baseDN "cn=monitor"  \
-D"cn=directory manager" gauge-name="Replication Latency (Milliseconds)"

dn: cn=Gauge Replication Latency (Milliseconds),cn=monitor
objectClass: top
```

```
objectClass: ds-monitor-entry
objectClass: ds-numeric-gauge-monitor-entry
objectClass: ds-gauge-monitor-entry
objectClass: extensibleObject
cn:          Gauge Replication Latency (Milliseconds)
gauge-name:  Replication Latency (Milliseconds)
resource:
severity:    indeterminate
summary:     The value of gauge Replication Latency (Milliseconds) could not
             be determined. The severity is INDETERMINATE, having assumed
             this severity Tue Aug 26 15:42:40 CDT 2014
error-message: No entries were found under cn=monitor having object
               class ds-replica-monitor-entry
              …
```

Managing SCIM Servlet Extensions

The PingDirectory Server provides two System for Cross-domain Identity Management (SCIM) servlet extensions to facilitate moving users to, from, and between cloud-based Software-as-a-Service (SaaS) applications in a secure, fast, and simple way. The SCIM 1.1 implementation is provided by the servlet extension named SCIM. The SCIM 2 implementation is provided by the servlet extension named SIMC2.

SCIM 1.1 and 2.0 servlet extensions management

The PingDirectory Server provides two System for Cross-domain Identity Management (SCIM) servlet extensions to facilitate moving users to, from, and between cloud-based Software-as-a-Service (SaaS) applications in a secure, fast, and simple way. SCIM is an alternative to LDAP, allowing identity data provisioning between cloud-based applications over HTTPS. One servelt implements SCIM 1.1 and the other servlet implements SCIM 2.

This section describes fundamental SCIM concepts and provides information on configuring SCIM on your server.

Overview of SCIM 1.1 fundamentals

Understanding the basic concepts of SCIM can help you use the SCIM extension to meet your deployment needs. SCIM allows you to:

- **Provision identities.** Through the API, you have access to the basic create, read, update, and delete functions, as well as other special functions.
- **Provision groups.** SCIM also allows you to manage groups.
- **Interoperate using a common schema.** SCIM provides a well-defined, platform-neutral user and group schema, as well as a simple mechanism to extend it.

The SCIM extension implements the 1.1 version of the SCIM specification. Familiarize yourself with this specification to help you understand and make efficient use of the SCIM extension and the SCIM SDK. The SCIM specifications are located on the Simplecloud website.

ⓘ **Note:** SCIM 1.1 will be deprecated in a future release.

Summary of SCIM 1.1 protocol support

PingDirectory Server supports all required features of the SCIM 1.1 protocol and most optional features. The following table describes SCIM features and whether they are supported.

SCIM Protocol Support

SCIM Feature	Supported
Etags	Yes
JSON	Yes
XML*	Yes
Authentication/Authorization	Yes, via HTTP basic authentication or OAuth 2.0 bearer tokens
Service Provider Configuration	Yes
Schema	Yes
User resources	Yes
Group resources	Yes
User-defined resources	Yes
Resource retrieval via GET	Yes
List/query resources	Yes
Query filtering*	Yes
Query result sorting*	Yes
Query result pagination*	Yes (Directory Server, not Directory Proxy Server)
Resource updates via PUT	Yes
Partial resource updates via PATCH*	Yes
Resource deletes via DELETE	Yes
Resource versioning*	Yes (requires configuration for updated servers)
Bulk*	Yes
HTTP method overloading	Yes
Raw LDAP Endpoints**	Yes

* denotes an optional feature of the SCIM protocol.

** denotes a PingDirectory Server extension to the basic SCIM functionality.

About the Identity Access API

The PingDirectory Server, PingDirectoryProxy Server, and PingDataSync Server support an extension to the SCIM 1.1 standard called the Identity Access API. The Identity Access API provides an alternative to LDAP by supporting CRUD (create, read, update, and delete) operations to access directory server data over an HTTP connection.

SCIM 1.1 and the Identity Access API are provided as a unified service through the SCIM HTTP Servlet Extension. The SCIM HTTP Servlet Extension can be configured to only enable core SCIM resources (e.g., 'Users' and 'Groups'), only LDAP object classes (e.g., `top`, `domain`, `inetOrgPerson`, or `groupOfUniqueNames`), or both. Because SCIM and the Identity Access API have different schemas, if both are enabled, there may be two representations with different schemas for any resources defined in the `scim-resources.xml` file: the SCIM representation and the raw LDAP representation. Likewise, because resources are exposed by an LDAP object class, and because these are hierarchical (e.g., `top` --> `person` --> `organizationalPerson` --> `inetOrgPerson`, etc.), a client application can access an entry in multiple ways due to the different paths/URIs to a given resource.

This chapter provides information on configuring the SCIM and the Identity Access API services on the PingDirectory Server.

Configuring SCIM 1.1

This section discusses details about the PingDirectory Server implementation of the SCIM protocol. Before reading this chapter, familiarize yourself with the SCIM Protocol specification, available on the Simplecloud website.

Creating your own SCIM 1.1 application

The System for Cross-domain Identity Management (SCIM) is an open initiative designed to make moving identity data to, from, and between clouds standard, secure, fast, and easy. The SCIM SDK is a pre-packaged collection of libraries and extensible classes that provides developers with a simple, concrete API to interact with a SCIM service provider.

The SCIM 1.1 SDK is available for download at https://github.com/pingidentity/scim.

ⓘ **Note:** The value of a read-only SCIM attribute can be set by a POST operation if the SCIM attribute is a custom attribute in the **`scim-resource.xml`** config file, but not if the SCIM attribute is a core SCIM attribute.

Configuring the SCIM 1.1 servlet extension

The Directory Proxy Server provides a default SCIM HTTP Servlet Extension that can be enabled and configured using a **`dsconfig`** batch script located in the `config` directory. The script runs a series of commands that enables an HTTPS Connection Handler, increases the level of detail logged by the HTTP Detailed Access log publisher, and adds access controls to allow access to LDAP controls used by the SCIM Servlet Extension. There are additional optional configurations (e.g., changing the log format, enable `entryDN` virtual attribute and using VLV indexes) that you can make by altering the `dsconfig` batch script.

The SCIM resource mappings are defined by the `scim-resources.xml` file located in the `config` directory. This file defines the SCIM schema and maps it to the LDAP schema. This file can be customized to define and expose deployment specific resources. See *Managing the SCIM Schema* for more information.

The following procedures show how to configure SCIM on the server. The first example procedure shows the steps to manually configure SCIM without using the script. The second example procedure uses the `dsconfig` batch script to configure SCIM.

Configuring SCIM manually

About this task

The following example procedure assumes that you have configured the Directory Server using the default settings, which means that SSL and the HTTPS Connection Handler have not been configured. The example also shows the **`dsconfig`** non-interactive commands. You can easily use the **`dsconfig`** interactive commands, which uses a menu-driven interface. If you use the **`dsconfig`** interactive, you must change to the Standard or Advanced object menu to access many of these configuration settings.

Steps

1. Set up your certificates. Follow the examples shown in the section *Managing Certificates*. You should have a keystore and truststore set up in the `config` directory. Make sure that the keystore.pin and truststore.pin are set.
2. Enable the key manager provider. The key manager provider accesses the certificate during the SSL handshaking process. If running **`dsconfig`** interactive, open the main menu, select "Key Manager

Provider" -> "View and edit an existing Key Manager Provider" -> "JKS" (or the type of certificate you are working with) -> "enabled" and then set the value to "true". Click "finish" to save the setting.

```
$ bin/dsconfig create-key-manager-provider \
  --provider-name JKS \
  --type file-based --set enabled:true
  --set key-store-file:config/keystore \
  --set key-store-type:JKS
  --set key-store-pin-file:config/keystore.pin
```

3. Enable the trust manager provider. The trust manager provider determines if a presented certificate can be trusted. If running **dsconfig** interactive, open the main menu, select "Trust Manager Provider" -> "View and edit an existing Trust Manager Provider" -> "JKS" (or the type of certificate you are working with) -> "enabled" and then set the value to "true". Click "finish" to save the setting.

```
$ bin/dsconfig create-trust-manager-provider
  --provider-name JKS \
  --type file-based --set enabled:true \
  --set trust-store-file:config/truststore \
  --set trust-store-type:JKS \
  --set trust-store-pin-file:config/truststore.pin
```

4. Configure the HTTPS Connection Handler. If not already created, this command creates and enables the connection handler, specifies the SCIM HTTP servlet extension and sets the listen port to a port of your choice, in this example, use 8443. The command also specifies the type of key manager and trust manager providers and sets the log publisher to "HTTP Detailed Access." If running **dsconfig** interactive, open the main menu, select "Connection Handler" -> "View and edit an existing Connection Handler" -> "HTTPS Connection Handler". Change the parameters to match your setup, and then, click "finish" to save the setting.

```
$ bin/dsconfig create-connection-handler \
  --handler-name "HTTPS Connection Handler" \
  --type http --set enabled:true \
  --set listen-port:8443 \
  --set use-ssl:true \
  --set http-servlet-extension:SCIM \
  --set "http-operation-log-publisher:HTTP Detailed Access" \
  --set key-manager-provider:JKS --set trust-manager-provider:JKS
```

If the HTTPS Connection Handler already exists, use the following:

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

5. Turn on the connection handler.

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

6. Add access controls to allow access to LDAP controls used by the SCIM Servlet Extension. These controls are the Post-Read Request Control (1.3.6.1.1.13.2), Server-Side Sort Request Control (1.2.840.113556.1.4.473), Simple Paged Results Control (1.2.840.113556.1.4.319), and Virtual List View Request Control (2.16.840.1.113730.3.4.9). We recommend using the following command to add access control instructions, rather than its **dsconfig** interactive equivalent.

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

7. Add access controls to allow read access to operational attributes used by the SCIM Servlet Extension. We recommend using the following non-interactive command to add access control instructions, rather than its **`dsconfig`** interactive equivalent.

```
$ bin/dsconfig set-access-control-handler-prop \
  --add 'global-aci:(targetattr="entryUUID || entryDN || ds-entry-unique-id
 ||
    createTimestamp || modifyTimestamp")
    (version 3.0;acl "Authenticated read access to operational attributes \
    used by the SCIM servlet extension"; allow (read,search,compare)
    userdn="ldap:///all";)'
```

8. Optional. The SCIM HTTP Connection Handler automatically uses a detailed HTTP log publisher, which is implemented in a proprietary format. If you need a standard W3C common log format publisher, enter the following command. If running **`dsconfig`** interactive, open the main menu, select "Log Publisher" -> "Create a new Log Publisher" -> "new Log Publisher created from scratch" -> "File Based Access Log Publisher", enter the parameters to match your setup, and then, click "finish" to save the setting. Go back to the main menu, select "Connection Handler" -> "HTTPS Connection Handler", and then add "HTTP Common Access" to the `http-operation-log-publisher` property. Click "finish" to save the setting.

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

$ bin/dsconfig set-connection-handler-prop \
  --handler-name "HTTPS Connection Handler" \
  --add "http-operation-log-publisher:HTTP Common Access"
```

9. Optional. To support searching or filtering by DN using the Identity Access API, you can enable the entryDN virtual attribute. If running **`dsconfig`** interactive, open the main menu, select "Virtual Attribute" -> "View and edit an existing Virtual Attribute" -> "Entry DN", and then change the enabled property to "true". Click "finish" to save the setting.

```
$ bin/dsconfig set-virtual-attribute-prop --name entryDN --set enabled:true
```

10. Optional. To support pagination, create some Virtual List View indexes. If running **`dsconfig`** interactive, open the main menu, select "Local DB VLV Index" -> "Create a new Local DB VLV Index" and then enter the properties needed for your setup. Click "finish" to save the setting. Repeat again for the "ascending-sn" index. Then, run the **`rebuild-index`** command to let the VLV Indexes take effect.

```
$ bin/dsconfig create-local-db-vlv-index --backend-name userRoot \
  --index-name ascending-uid --set base-dn:dc=example,dc=com \
  --set scope:whole-subtree \
  --set "filter:(objectclass=inetorgperson)" \
  --set "sort-order:+uid"

$ bin/dsconfig create-local-db-vlv-index --backend-name userRoot \
  --index-name ascending-sn \
  --set base-dn:dc=example,dc=com \
  --set scope:whole-subtree \
  --set "filter:(objectclass=inetorgperson)" \
  --set "sort-order:+sn"

$ bin/rebuild-index --baseDN dc=example,dc=com \
  --index vlv.ascending-uid \
  --index vlv.ascending-sn
```

Enabling resource versioning

About this task

Resource versioning is enabled by default in new installations. Upgraded servers that had SCIM enabled need additional configuration to enable resource versioning.

Steps

1. Enable the **`ds-entry-checksum`** virtual attribute.

```
$ bin/dsconfig set-virtual-attribute-prop \
    --name ds-entry-checksum \
    --set enabled:true
```

2. Remove any existing access controls required by SCIM for read access to operational attributes:

```
$ bin/dsconfig set-access-control-handler-prop \
    --remove 'global-aci:(targetattr="entryUUID || entryDN || ds-entry-
unique-id || createTimestamp || ds-create-time || modifyTimestamp || ds-
update-time")(version 3.0;acl "Authenticated read access to operational
 attributes used by the SCIM servlet extension"; allow (read,search,compare)
 userdn="ldap:///all"'
```

3. Add new access controls required by SCIM for read access to operational attributes with the addition of the **`ds-entry-checksum`**:

```
$ bin/dsconfig set-access-control-handler-prop \
    --add  'global-aci:(targetattr="entryUUID || entryDN || ds-entry-unique-
id || createTimestamp || ds-create-time || modifyTimestamp || ds-update-
time || ds-entry-checksum")(version 3.0;acl "Authenticated read access
 to operational attributes used by the SCIM servlet extension"; allow
 (read,search,compare) userdn="ldap:///all"'
```

4. Enable SCIM resource versioning using the entry checksum virtual attribute:

```
$ bin/dsconfig set-http-servlet-extension-prop \
    --extension-name SCIM \
    --set entity-tag-ldap-attribute:ds-entry-checksum
```

   If enabled, the value of the **`ds-entry-checksum`** attribute is returned as the **`ETag`** header value when accessing the resource through SCIM, and is checked against the **`If-Match`** header when updating the resource. When accessing the resource through LDAP, use the **`ds-entry-checksum`** attribute instead.

Configuring the SCIM servlet extension using the batch script

About this task

The following example procedure assumes that you have set up your certificates, keystore, and truststore

Steps

1. Open the `<server-root>/config/scim-config-ds.dsconfig` script in a text editor.
2. For the optional elements (W3C common log, filtering by DN, and VLV Indexes, remove the comment (#) symbol on the **`dsconfig`** commands. Save the file when finished editing.
3. To enable the SCIM servlet extension, run the **`dsconfig`** batch file. Remember to include the bind parameters.

```
$ bin/dsconfig --batch-file config/scim-config-ds.dsconfig
```

SCIM 1.1 servlet extension authentication

The SCIM 1.1 servlet supports authentication using either the HTTP Basic authentication scheme, or OAuth 2.0 bearer tokens. When authenticating using HTTP Basic authentication, the SCIM 1.1 servlet attempts to correlate the user name component of the Authorization header to a DN in the Directory Server. If the user name value cannot be parsed directly as a DN, it is correlated to a DN using an Identity Mapper. The DN is then used in a simple bind request to verify the password.

In deployments that use an OAuth authorization server, the SCIM 1.1 extension can be configured to authenticate requests using OAuth bearer tokens. The SCIM 1.1 extension supports authentication with OAuth 2.0 bearer tokens (per RFC 6750) using an OAuth Token Handler Server SDK Extension. Because the OAuth 2.0 specification does not specify how contents of a bearer token are formatted, PingDirectory Server provides the token handler API to decode incoming bearer tokens and extract or correlate associated authorization DNs.

Neither HTTP Basic authentication nor OAuth 2.0 bearer token authentication are secure unless SSL is used to encrypt the HTTP traffic.

Configuring basic authentication using an identity mapper

About this task

By default, the SCIM servlet is configured to use the Exact Match Identity Mapper, which matches against the **`uid`** attribute. In this example, an alternate Identity Mapper is created so that clients can authenticate using **`cn`** values.

Steps

1. Create a new Identity Mapper that uses a match attribute of cn.

```
$ bin/dsconfig create-identity-mapper \
  --mapper-name "CN Identity Mapper" \
  --type exact-match \
  --set enabled:true \
  --set match-attribute:cn
```

2. Configure the SCIM servlet to use the new Identity Mapper.

```
$ bin/dsconfig set-http-servlet-extension-prop \
  --extension-name SCIM \
  --set "identity-mapper:CN Identity Mapper"
```

Enabling OAuth authentication

About this task

To enable OAuth authentication, you need to create an implementation of the `OAuthTokenHandler` using the API provided in the Server SDK. For details on creating an `OAuthTokenHandler` extension, see the Server SDK documentation.

Steps

1. Install your OAuth token handler on the server using **`dsconfig`**.

```
$ bin/dsconfig create-oauth-token-handler \
  --handler-name ExampleOAuthTokenHandler \
  --type third-party \
  --set extension-
class:com.unboundid.directory.sdk.examples.ExampleOAuthTokenHandler
```

2. Configure the SCIM servlet extension to use it as follows:

```
$ bin/dsconfig set-http-servlet-extension-prop \
```

```
  --extension-name SCIM \
  --set oauth-token-handler:ExampleOAuthTokenHandler
```

Verifying the SCIM 1.1 servlet extension configuration

About this task

You can verify the configuration of the SCIM 1.1 extension by navigating to a SCIM resource URL via the command line or through a browser window.

To verify the SCIM servlet extension configuration:

Steps

- Run **`curl`** to verify that the SCIM extension is running. The `-k` (or `--insecure`) option is used to turn off curl's verification of the server certificate, since the example Directory Server is using a self-signed certificate.

```
$ curl -u "cn=Directory Manager:password" \
-k "https://localhost:8443/scim/ServiceProviderConfigs"

{"schemas":["urn:scim:schemas:core:1.0"],"id":"urn:scim:schemas:core:1.0",
"patch":{"supported":true},"bulk":{"supported":true,"maxOperations":10000,
"maxPayloadSize":10485760},"filter":{"supported":true,"maxResults":100},
"changePassword":{"supported":true},"sort":{"supported":true},
"etag":{"supported":false},"authenticationSchemes":[{"name":"HttpBasic",
"description":"The HTTP Basic Access Authentication scheme. This scheme is
not considered to be a secure method of user authentication (unless used in
conjunction with some external secure system such as SSL), as the user
name and password are passed over the network as cleartext.","specUrl":
"http://www.ietf.org/rfc/rfc2617","documentationUrl":
"http://en.wikipedia.org/wiki/Basic_access_authentication"}]}
```

- If the user ID is a valid DN (such as `cn=Directory Manager`), the SCIM extension authenticates by binding to the Directory Server as that user. If the user ID is not a valid DN, the SCIM extension searches for an entry with that `uid` value, and binds to the server as that user. To verify authentication to the server as the user with the `uid` of `user.0`, run the following command:

```
$ curl -u "user.0:password" \
  -k "https://localhost:8443/scim/ServiceProviderConfigs"
```

Configuring the Identity Access API

Once you have run the `<server-root>/config/scim-config-ds.dsconfig` script, the resources defined in the `scim-resources.xml` will be available as well as the Identity Access API. However, to allow SCIM access to the raw LDAP data, you must set a combination of configuration properties on the SCIM Servlet Extension using the **`dsconfig`** tool.

- **include-ldap-objectclass**. Specifies a multi-valued property that lists the object classes for entries that will be exposed. The object class used here will be the one that clients need to use when referencing Identity Access API resources. This property allows the special value "*" to allow all object classes. If "*" is used, then the SCIM servlet uses the same case used in the Directory Server LDAP Schema.
- **exclude-ldap-objectclass**. Specifies a multi-valued property that lists the object classes for entries that will not be exposed. When this property is specified, all object classes will be exposed except those in this list.
- **include-ldap-base-dn**. Specifies a multi-valued property that lists the base DNs that will be exposed. If specified, only entries under these base DNs will be accessible. No parent-child relationships in the DNs are allowed here.
- **exclude-ldap-base-dn**. Specifies a multi-valued property that lists the base DNs that will not be exposed. If specified, entries under these base DNs will not be accessible. No parent-child relationships in the DNs are allowed here.

Using a combination of these properties, SCIM endpoints will be available for all included object classes, just as if they were SCIM Resources defined in the `scim-resources.xml` file.

Configuring the Identity Access API

Steps

1. Ensure that you have run the `scim-config-ds.dsconfig` script to configure the SCIM interface. Be sure to enable the entryDN virtual attribute. See the Configure SCIM section for more information.
2. Set a combination of properties to allow the SCIM clients access to the raw LDAP data: `include-ldap-objectclass`, `exclude-ldap-objectclass`, `include-ldap-base-dn`, or `exclude-ldap-base-dn`.

```
$ bin/dsconfig set-http-servlet-extension-prop \
  --extension-name SCIM --set 'include-ldap-objectclass:*' \
  --set include-ldap-base-dn:ou=People,dc=example,dc=com
```

   The SCIM clients now have access to the raw LDAP data via LDAP object class-based resources as well as core SCIM resources as defined in the `scim.resource.xml` file.

Disabling core SCIM resources

Steps

1. Open the `config/scim-resources.xml` file, and comment out or remove the `<resource>` elements that you would like to disable.
2. Disable and re-enable the HTTP Connection Handler, or restart the server to make the changes take effect. In general, changing the `scim-resources.xml` file requires a HTTP Connection Handler restart or server restart.

> ⓘ **Note:** When making other changes to the SCIM configuration by modifying the SCIM HTTP Servlet Extension using **`dsconfig`**, the changes take effect immediately without any restart required.

Verifying the Identity Access API configuration

Steps

- Perform a curl request to verify the Identity Access API configuration.

```
$ curl -k -u "cn=directory manager:password" \
  -H "Accept: application/json" \
  "https://example.com/top/56c9fd6b-f870-35ef-9959-691c783b7318?
     attributes=entryDN,uid,givenName,sn,entryUUID"
     {"schemas":
["urn:scim:schemas:core:1.0","urn:unboundid:schemas:scim:ldap:1.0"],
      "id":"56c9fd6b-f870-35ef-9959-691c783b7318",
      "meta":{"lastModified":"2013-01-11T23:38:26.489Z",
      "location":"https://example.com:443/v1/top/56c9fd6b-
f870-35ef-9959-691c783b7318"},
      "urn:unboundid:schemas:scim:ldap:1.0":{"givenName":["Rufus"],"uid":
["user.1"],
      "sn":["Firefly"],"entryUUID":["56c9fd6b-f870-35ef-9959-691c783b7318"],
      "entrydn":"uid=user.1,ou=people,dc=example,dc=com"}}
```

Monitoring the SCIM servlet extension

The SCIM SDK provides a command-line tool, `scim-query-rate`, that measures the SCIM query performance for your extension. The SCIM extension also exposes monitoring information for each SCIM resource, such as the number of successful operations per request, the number of failed operations per request, the number of operations with XML or JSON to and from the client. Finally, the Directory

Server automatically logs SCIM-initiated LDAP operations to the default File-based Access Logger. These operations will have an `origin='scim'` attribute to distinguish them from operations initiated by LDAP clients. You can also create custom logger or request criteria objects that can track incoming HTTP requests, which the SCIM extension rewrites as internal LDAP operations.

Testing SCIM query performance

You can use the `scim-query-rate` tool, provided in the SCIM SDK, to test query performance, by performing repeated resource queries against the SCIM server.

The `scim-query-rate` tool performs searches using a query filter or can request resources by ID. For example, you can test performance by using a filter to query randomly across a set of one million users with eight concurrent threads. The user resources returned to the client in this example is in XML format and includes the `userName` and `name` attributes.

```
scim-query-rate --hostname server.example.com --port 80 \
--authID admin --authPassword password --xml \
--filter 'userName eq "user.[1-1000000]"' --attribute userName \
--attribute name --numThreads 8
```

You can request resources by specifying a resource ID pattern using the `--resourceID` argument as follows:

```
scim-query-rate --hostname server.example.com --port 443 \
--authID admin --authPassword password --useSSL --trustAll\
--resourceName User \
--resourceID 'uid=user.[1-150000],ou=people,dc=example,dc=com'
```

The `scim-query-rate` tool reports the error `"java.net.SocketException: Too many open files"` if the open file limit is too low. You can increase the open file limit to increase the number of file descriptors.

Monitoring resources using the SCIM extension

The monitor provider exposes the following information for each resource:

- Number of successful operations per request type (such as GET, PUT, and POST).
- Number of failed operations and their error codes per request type.
- Number of operations with XML or JSON from client.
- Number of operations that sent XML or JSON to client.

In addition to the information about the user-defined resources, monitoring information is also generated for the schema, service provider configuration, and monitor resources. The attributes of the monitor entry are formatted as follows:

```
{resource name}-resource-{request type}-{successful or error status code}
```

You can search for one of these monitor providers using an **`ldapsearch`** such as the following:

```
$ bin/ldapsearch --port 1389 bindDN uid=admin,dc=example,dc=com \
  --bindPassword password --baseDN cn=monitor \
  --searchScope sub "(objectclass=scim-servlet-monitor-entry)"
```

For example, the following monitor output was produced by a test environment with three distinct SCIM servlet instances, Aleph, Beth, and Gimel. Note that the first instance has a custom resource type called host.

```
$ bin/ldapsearch --baseDN cn=monitor \
  '(objectClass=scim-servlet-monitor-entry)'
dn: cn=SCIM Servlet (SCIM HTTP Connection Handler),cn=monitor
objectClass: top
objectClass: ds-monitor-entry
objectClass: scim-servlet-monitor-entry
```

```
objectClass: extensibleObject
cn: SCIM Servlet (SCIM HTTPS Connection Handler) [from
  ThirdPartyHTTPServletExtension:SCIM (Aleph)]
ds-extension-monitor-name: SCIM Servlet (SCIM HTTPS Connection Handler)
ds-extension-type: ThirdPartyHTTPServletExtension
ds-extension-name: SCIM (Aleph)
version: 1.2.0
build: 20120105174457Z
revision: 820
schema-resource-query-successful: 8
schema-resource-query-401: 8
schema-resource-query-response-json: 16
user-resource-delete-successful: 1
user-resource-put-content-xml: 27
user-resource-query-response-json: 3229836
user-resource-put-403: 5
user-resource-put-content-json: 2
user-resource-get-401: 1
user-resource-put-response-json: 23
user-resource-get-response-json: 5
user-resource-get-response-xml: 7
user-resource-put-400: 2
user-resource-query-401: 1141028
user-resource-post-content-json: 1
user-resource-put-successful: 22
user-resource-post-successful: 1
user-resource-delete-404: 1
user-resource-query-successful: 2088808
user-resource-get-successful: 10
user-resource-put-response-xml: 6
user-resource-get-404: 1
user-resource-delete-401: 1
user-resource-post-response-json: 1
host-resource-query-successful: 5773268
host-resource-query-response-json: 11576313
host-resource-query-400: 3
host-resource-query-response-xml: 5
host-resource-query-401: 5788152
dn: cn=SCIM Servlet (SCIM HTTP Connection Handler),cn=monitor
objectClass: top
objectClass: ds-monitor-entry
objectClass: scim-servlet-monitor-entry
objectClass: extensibleObject
cn: SCIM Servlet (SCIM HTTPS Connection Handler) [from
  ThirdPartyHTTPServletExtension:SCIM (Beth)]
ds-extension-monitor-name: SCIM Servlet (SCIM HTTPS Connection
  Handler)
ds-extension-type: ThirdPartyHTTPServletExtension
ds-extension-name: SCIM (Beth)
version: 1.2.0
build: 20120105174457Z
revision: 820
serviceproviderconfig-resource-get-successful: 3
serviceproviderconfig-resource-get-response-json: 2
serviceproviderconfig-resource-get-response-xml: 1
schema-resource-query-successful: 8
schema-resource-query-401: 8
schema-resource-query-response-json: 16
group-resource-query-successful: 245214
group-resource-query-response-json: 517841
group-resource-query-400: 13711
group-resource-query-401: 258916
user-resource-query-response-json: 107876
user-resource-query-400: 8288
```

```
user-resource-get-400: 33
user-resource-get-response-json: 1041
user-resource-get-successful: 2011
user-resource-query-successful: 45650
user-resource-get-response-xml: 1003
user-resource-query-401: 53938
dn: cn=SCIM Servlet (SCIM HTTP Connection Handler),cn=monitor
objectClass: top
objectClass: ds-monitor-entry
objectClass: scim-servlet-monitor-entry
objectClass: extensibleObject
cn: SCIM Servlet (SCIM HTTPS Connection Handler) [from
  ThirdPartyHTTPServletExtension:SCIM (Gimel)]
ds-extension-monitor-name: SCIM Servlet (SCIM HTTPS Connection
  Handler)
ds-extension-type: ThirdPartyHTTPServletExtension
ds-extension-name: SCIM (Gimel)
version: 1.2.0
build: 20120105174457Z
revision: 820
schema-resource-query-successful: 1
schema-resource-query-401: 1
schema-resource-query-response-json: 2
user-resource-query-successful: 65
user-resource-get-successful: 4
user-resource-get-response-json: 6
user-resource-query-response-json: 132
user-resource-get-404: 2
user-resource-query-401: 67
```

About the HTTP log publishers

HTTP operations may be logged using either a Common Log File HTTP Operation Log Publisher or a Detailed HTTP Operation Log Publisher. The Common Log File HTTP Operation Log Publisher is a built-in log publisher that records HTTP operation information to a file using the W3C common log format. Because the W3C common log format is used, logs produced by this log publisher can be parsed by many existing web analysis tools.

Log messages are formatted as follows:

- IP address of the client.
- RFC 1413 identification protocol. The Ident Protocol is used to format information about the client.
- The user ID provided by the client in an Authorization header, which is typically available server-side in the REMOTE_USER environment variable. A dash appears in this field if this information is not available.
- A timestamp, formatted as '"['dd/MM/yyyy:HH:mm:ss Z']"'
- Request information, with the HTTP method followed by the request path and HTTP protocol version.
- The HTTP status code value.
- The content size of the response body in bytes. This number does not include the size of the response headers.

The HTTP Detailed Access Log Publisher provides more information than the common log format in a format that is familiar to administrators who use the File-Based Access Log Publisher.

The HTTP Detailed Access Log Publisher generates log messages such as the following. The lines have been wrapped for readability.

```
[15/Feb/2012:21:17:04 -0600] RESULT requestID=10834128
from="10.2.1.114:57555" method="PUT"
url="https://10.2.1.129:443/Aleph/Users/6272c691-
38c6-012f-d227-0dfae261c79e" authorizationType="Basic"
requestContentType="application/json" statusCode=200
```

```
etime=3.544 responseContentLength=1063
redirectURI="https://server1.example.com:443/Aleph/Users/6272c691-38c6-012f-
d227-0dfae261c79e"
responseContentType="application/json"
```

In this example, only default log publisher properties are used. Though this message is for a RESULT, it contains information about the request, such as the client address, the request method, the request URL, the authentication method used, and the Content-Type requested. For the response, it includes the response length, the redirect URI, the Content-Type, and the HTTP status code.

You can modify the information logged, including adding request parameters, cookies, and specific request and response headers. For more information, refer to the **`dsconfig`** command-line tool help.

Configuring advanced SCIM 1.1 extension features

The following sections show how to configure advanced SCIM 1.1 servlet extension features, such as bulk operation implementation, mapping SCIM resource IDs, and transformations.

Managing the SCIM 1.1 schema

This section describes the SCIM schema and provides information on how to map LDAP schema to the SCIM resource schema.

About the SCIM schema

SCIM provides a common user schema and extension model, making it easier to interoperate with multiple Service Providers. The core SCIM schema defines a concrete schema for user and group resources that encompasses common attributes found in many existing schemas.

Each attribute is defined as either a single attribute, allowing only one instance per resource, or a multi-valued attribute, in which case several instances may be present for each resource. Attributes may be defined as simple, name-value pairs or as complex structures that define sub-attributes.

While the SCIM schema follows an object extension model similar to object classes in LDAP, it does not have an inheritance model. Instead, all extensions are additive, similar to LDAP Auxiliary Object Classes.

Mapping the LDAP schema to the SCIM resource schema

The resources configuration file is an XML file that is used to define the SCIM resource schema and its mapping to LDAP schema. The default configuration of the `scim-resources.xml` file provides definitions for the standard SCIM Users and Groups resources, and mappings to the standard LDAP `inetOrgPerson` and `groupOfUniqueNames` object classes.

The default configuration may be customized by adding extension attributes to the Users and Groups resources, or by adding new extension resources. The resources file is composed of a single `<resources>` element, containing one or more `<resource>` elements.

For any given SCIM resource endpoint, only one `<LDAPAdd>` template can be defined, and only one `<LDAPSearch>` element can be referenced. If entries of the same object class can be located under different subtrees or base DNs of the Directory Server, then a distinct SCIM resource must be defined for each unique entry location in the Directory Information Tree. This can be implemented in many ways. For example:

- Create multiple SCIM servlets, each with a unique `scim-resources.xml` configuration, and each running under a unique HTTP connection handler.
- Create multiple SCIM servlets, each with a unique `scim-resources.xml` configuration, each running under a single, shared HTTP connection handler, but each with a unique context path.

Note that LDAP attributes are allowed to contain characters that are invalid in XML (because not all valid UTF-8 characters are valid XML characters). The easiest and most-correct way to handle this is to make sure that any attributes that may contain binary data are declared using "dataType=binary" in the `scim-resources.xml` file. Likewise, when using the Identity Access API make sure that the underlying LDAP schema uses the Binary or Octet String attribute syntax for attributes which may contain binary data. This

will cause the server to automatically base64-encode the data before returning it to clients and will also make it predictable for clients because they can assume the data will always be base64-encoded.

However, it is still possible that attributes that are not declared as binary in the schema may contain binary data (or just data that is invalid in XML), and the server will always check for this before returning them to the client. If the client has set the content-type to XML, then the server may choose to base64-encode any values which are found to include invalid XML characters. When this is done, a special attribute is added to the XML element to alert the client that the value is base64-encoded. For example:

```
<scim:value base64Encoded="true">AAABPB0EBZc=</scim:value>
```

The remainder of this section describes the mapping elements available in the `scim-resources.xml` file.

About the `resource` element

A `resource` element has the following XML attributes:

- `schema`: a required attribute specifying the SCIM schema URN for the resource. Standard SCIM resources already have URNs assigned for them, such as `urn:scim:schemas:core:1.0`. A new URN must be obtained for custom resources using any of the standard URN assignment methods.
- `name`: a required attribute specifying the name of the resource used to access it through the SCIM REST API.
- `mapping`: a custom Java class that provides the logic for the resource mapper. This class must extend the `com.unboundid.scim.ldap.ResourceMapper` class.

A resource element contains the following XML elements in sequence:

- `description`: a required element describing the resource.
- `endpoint`: a required element specifying the endpoint to access the resource using the SCIM REST API.
- `LDAPSearchRef`: a mandatory element that points to an `LDAPSearch` element. The `LDAPSearch` element allows a SCIM query for the resource to be handled by an LDAP service and also specifies how the SCIM resource ID is mapped to the LDAP server.
- `LDAPAdd`: an optional element specifying information to allow a new SCIM resource to be added through an LDAP service. If the element is not provided then new resources cannot be created through the SCIM service.
- `attribute`: one or more elements specifying the SCIM attributes for the resource.

About the `attribute` element

An `attribute` element has the following XML attributes:

- `schema`: a required attribute specifying the schema URN for the SCIM attribute. If omitted, the schema URN is assumed to be the same as that of the enclosing resource, so this only needs to be provided for SCIM extension attributes. Standard SCIM attributes already have URNs assigned for them, such as urn:scim:schemas:core:1.0. A new URN must be obtained for custom SCIM attributes using any of the standard URN assignment methods.
- `name`: a required attribute specifying the name of the SCIM attribute.
- `readOnly`: an optional attribute indicating whether the SCIM sub-attribute is not allowed to be updated by the SCIM service consumer. The default value is false.
- `required`: an optional attribute indicating whether the SCIM attribute is required to be present in the resource. The default value is false.

An attribute element contains the following XML elements in sequence:

- `description`: a required element describing the attribute. Then just one of the following elements:
  - `simple`: specifies a simple, singular SCIM attribute.
  - `complex`: specifies a complex, singular SCIM attribute.
  - `simpleMultiValued`: specifies a simple, multi-valued SCIM attribute.
  - `complexMultiValued`: specifies a complex, multi-valued SCIM attribute.

About the `simple` element

A `simple` element has the following XML attributes:

- `dataType`: a required attribute specifying the simple data type for the SCIM attribute. The following values are permitted: binary, boolean, dateTime, decimal, integer, string.
- `caseExact`: an optional attribute that is only applicable for string data types. It indicates whether comparisons between two string values use a case-exact match or a case-ignore match. The default value is false.

A simple element contains the following XML element:

- `mapping`: an optional element specifying a mapping between the SCIM attribute and an LDAP attribute. If this element is omitted, then the SCIM attribute has no mapping and the SCIM service ignores any values provided for the SCIM attribute.

About the `complex` element

The `complex` element does not have any XML attributes. It contains the following XML element:

- `subAttribute`: one or more elements specifying the sub-attributes of the complex SCIM attribute, and an optional mapping to LDAP. The standard `type`, `primary`, and `display` sub-attributes do not need to be specified.

About the `simpleMultivalued` element

A `simpleMultiValued` element has the following XML attributes:

- `childName`: a required attribute specifying the name of the tag that is used to encode values of the SCIM attribute in XML in the REST API protocol. For example, the tag for the standard emails SCIM attribute is email.
- `dataType`: a required attribute specifying the simple data type for the plural SCIM attribute (i.e. the data type for the value sub-attribute). The following values are permitted: `binary`, `boolean`, `dateTime`, `integer`, `string`.
- `caseExact`: an optional attribute that is only applicable for string data types. It indicates whether comparisons between two string values use a case-exact match or a case-ignore match. The default value is false.

A simpleMultiValued element contains the following XML elements in sequence:

- `canonicalValue`: specifies the values of the type sub-attribute that is used to label each individual value, and an optional mapping to LDAP.
- `mapping`: an optional element specifying a default mapping between the SCIM attribute and an LDAP attribute.

About the `complexMultiValued` element

A `complexMultiValued` element has the following XML attribute:

- `tag`: a required attribute specifying the name of the tag that is used to encode values of the SCIM attribute in XML in the REST API protocol. For example, the tag for the standard addresses SCIM attribute is address.

A `complexMultiValued` element contains the following XML elements in sequence:

- `subAttribute`: one or more elements specifying the sub-attributes of the complex SCIM attribute. The standard type, primary, and display sub-attributes do not need to be specified.

- `canonicalValue`: specifies the values of the type sub-attribute that is used to label each individual value, and an optional mapping to LDAP.

About the `subAttribute` element

A `subAttribute` element has the following XML attributes:

- `name`: a required element specifying the name of the sub-attribute.
- `readOnly`: an optional attribute indicating whether the SCIM sub-attribute is not allowed to be updated by the SCIM service consumer. The default value is false.
- `required`: an optional attribute indicating whether the SCIM sub-attribute is required to be present in the SCIM attribute. The default value is false.
- `dataType`: a required attribute specifying the simple data type for the SCIM sub-attribute. The following values are permitted: binary, boolean, dateTime, integer, string.
- `caseExact`: an optional attribute that is only applicable for string data types. It indicates whether comparisons between two string values use a case-exact match or a case-ignore match. The default value is false.

A `subAttribute` element contains the following XML elements in sequence:

- `description`: a required element describing the sub-attribute.
- `mapping`: an optional element specifying a mapping between the SCIM sub-attribute and an LDAP attribute. This element is not applicable within the complexMultiValued element.

About the `canonicalValue` element

A `canonicalValue` element has the following XML attribute:

- `name`: specifies the value of the type sub-attribute. For example, work is the value for emails, phone numbers and addresses intended for business purposes.

A `canonicalValue` element contains the following XML element:

- `subMapping`: an optional element specifying mappings for one or more of the sub-attributes. Any sub-attributes that have no mappings will be ignored by the mapping service.

About the `mapping` element

A `mapping` element has the following XML attributes:

- `ldapAttribute`: A required element specifying the name of the LDAP attribute to which the SCIM attribute or sub-attribute map.
- `transform`: An optional element specifying a transformation to apply when mapping an attribute value from SCIM to LDAP and vice-versa. The available transformations are described in the *Mapping LDAP Entries to SCIM Using the SCIM-LDAP API* section.

About the `subMapping` element

A `subMapping` element has the following XML attributes:

- `name`: a required element specifying the name of the sub-attribute that is mapped.
- `ldapAttribute`: a required element specifying the name of the LDAP attribute to which the SCIM sub-attribute maps.
- `transform`: an optional element specifying a transformation to apply when mapping an attribute value from SCIM to LDAP and vice-versa. The available transformations are described later. The available transformations are described in *Mapping LDAP Entries to SCIM Using the SCIM-LDAP API*.

About the `LDAPSearch` element

An `LDAPSearch` element contains the following XML elements in sequence:

- `baseDN`: a required element specifying one or more LDAP search base DNs to be used when querying for the SCIM resource.

- `filter`: a required element specifying an LDAP filter that matches entries representing the SCIM resource. This filter is typically an equality filter on the LDAP object class.
- `resourceIDMapping`: an optional element specifying a mapping from the SCIM resource ID to an LDAP attribute. When the element is omitted, the resource ID maps to the LDAP entry DN. Note The `LDAPSearch` element can be added as a top-level element outside of any <Resource> elements, and then referenced within them via an ID attribute.

> ⓘ **Note:** The `LDAPSearch` element can be added as a top-level element outside of any <Resource> elements, and then referenced within them via an ID attribute.

About the `resourceIDMapping` element

The `resourceIDMapping` element has the following XML attributes:

- `ldapAttribute`: a required element specifying the name of the LDAP attribute to which the SCIM resource ID maps.
- `createdBy`: a required element specifying the source of the resource ID value when a new resource is created by the SCIM consumer using a POST operation. Allowable values for this element include `scim-consumer`, meaning that a value must be present in the initial resource content provided by the SCIM consumer, or Directory Server, meaning that a value is automatically provided by the Directory Server (as would be the case if the mapped LDAP attribute is entryUUID).

The following example illustrates an `LDAPSearch` element that contains a `resourceIDMapping` element:

```
<LDAPSearch id="userSearchParams">
  <baseDN>ou=people,dc=example,dc=com</baseDN>
  <filter>(objectClass=inetOrgPerson)</filter>
  <resourceIDMapping ldapAttribute="entryUUID" createdBy="directory"/>
</LDAPSearch>
```

About the `LDAPAdd` element

An LDAPAdd element contains the following XML elements in sequence:

- `DNTemplate`: a required element specifying a template that is used to construct the DN of an entry representing a SCIM resource when it is created. The template may reference values of the entry after it has been mapped using `{ldapAttr}`, where `ldapAttr` is the name of an LDAP attribute.
- `fixedAttribute`: zero or more elements specifying fixed LDAP values to be inserted into the entry after it has been mapped from the SCIM resource.

About the `fixedAttribute` element

A `fixedAttribute` element has the following XML attributes:

- `ldapAttribute`: a required attribute specifying the name of the LDAP attribute for the fixed values.
- `onConflict`: an optional attribute specifying the behavior when the LDAP entry already contains the specified LDAP attribute. The value merge indicates that the fixed values should be merged with the existing values. The value overwrite indicates that the existing values are to be overwritten by the fixed values. The value preserve indicates that no changes should be made. The default value is merge.

A `fixedAttribute` element contains one or more `fixedValue` XML element, which specify the fixed LDAP values.

Validating the updated SCIM schema

The PingDirectory Server SCIM extension is bundled with an XML Schema document, `resources.xsd`, which describes the structure of a `scim-resources.xml` resource configuration file. After updating the resource configuration file, you should confirm that its contents are well-formed and valid using a tool such as `xmllint`.

For example, you could validate your updated file as follows:

```
$ xmllint --noout --schema resources.xsd scim-resources.xml
scim-resources.xml validates
```

Mapping SCIM resource IDs

The default `scim-resources.xml` configuration maps the SCIM resource ID to the LDAP `entryUUID` attribute. The `entryUUID` attribute, whose read-only value is assigned by the Directory Server, meets the requirements of the SCIM specification regarding resource ID immutability. However, configuring a mapping to the attribute may result in inefficient group processing, since LDAP groups use the entry DN as the basis of group membership. The resource configuration allows the SCIM resource ID to be mapped to the LDAP entry DN. However, the entry DN does not meet the requirements of the SCIM specification regarding resource ID immutability. LDAP permits entries to be renamed or moved, thus modifying the DN. Likewise, you can use the Identity Access API to change the value of an entry's RDN attribute, thereby triggering a MODDN operation.

A resource may also be configured such that its SCIM resource ID is provided by an arbitrary attribute in the request body during POST operations. This SCIM attribute must be mapped to an LDAP attribute so that the SCIM resource ID may be stored in the Directory Server. By default, it is the responsibility of the SCIM client to guarantee ID uniqueness. However, the UID Unique Attribute Plugin may be used by the Directory Server to enforce attribute value uniqueness. For information about the UID Unique Attribute Plugin, see "Working with the UID Unique Attribute plugin" in the *PingDirectory Server Administration Guide*.

ⓘ **Note:** Resource IDs may not be mapped to virtual attributes. For more information about configuring SCIM Resource IDs, see "About the `<resourceIDMapping>` Element".

Using pre-defined transformations

Transformations are required to change SCIM data types to LDAP syntax values. The following pre-defined transformations may be referenced by the transform XML attribute:

- `com.unboundid.scim.ldap.BooleanTransformation`. Transforms SCIM boolean data type values to LDAP Boolean syntax values and vice-versa.
- `com.unboundid.scim.ldap.GeneralizedTimeTransformation`. Transforms SCIM dateTime data type values to LDAP Generalized Time syntax values and vice-versa.
- `com.unboundid.scim.ldap.PostalAddressTransformation`. Transforms SCIM formatted address values to LDAP Postal Address syntax values and vice-versa. SCIM formatted physical mailing addresses are represented as strings with embedded new lines, whereas LDAP uses the $ character to separate address lines. This transformation interprets new lines in SCIM values as address line separators.
- `com.unboundid.scim.ldap.TelephoneNumberTransformation`. Transforms LDAP Telephone Number syntax (E.123) to RFC3966 format and vice-versa.

You can also write your own transformations using the SCIM API described in the following section.

Mapping LDAP entries to SCIM using the SCIM-LDAP API

In addition to the SCIM SDK, PingDirectoryProxy Server provides a library called SCIM-LDAP, which provides facilities for writing custom transformations and more advanced mapping.

You can add the SCIM-LDAP library to your project using the following dependency:

```
<dependency>
    <groupId>com.unboundid.product.scim</groupId>
    <artifactId>scim-ldap</artifactId>
    <version>1.5.0</version>
</dependency>
```

Create your custom transformation by extending the `com.unboundid.scim.ldap.Transformation` class. Place your custom transformation class in a jar file in the server's `lib` directory.

ⓘ **Note:** The Identity Access API automatically maps LDAP attribute syntaxes to the appropriate SCIM attribute types. For example, an LDAP DirectoryString is automatically mapped to a SCIM string.

SCIM authentication

SCIM requests to the LDAP endpoints will support HTTP Basic Authentication and OAuth2 Authentication using a bearer token. There is existing support for this feature in the Directory Server and the Directory Proxy Server using the OAuthTokenHandler API (i.e., via a Server SDK extension, which requires some technical work to implement).

Note that our implementation only supports the HTTP Authorization header for this purpose; we do not support the form-encoded body parameter or URI query parameter mechanisms for specifying the credentials or bearer token.

SCIM logging

The Directory Server already provides a detailed HTTP log publisher to capture the SCIM and HTTP request details. To be able to correlate this data to the internal LDAP operations that are invoked behind the scenes, the Access Log Publisher will use "`origin=scim`" in access log messages that are generated by the SCIM servlet.

For example, you will see a message for operations invoked by replication:

```
[30/Oct/2012:18:45:10.490 -0500] MODIFY REQUEST conn=-3 op=190 msgID=191
origin="replication" dn="uid=user.3,ou=people,dc=example,dc=com"
```

Likewise for SCIM messages, you will see a message like this:

```
[30/Oct/2012:18:45:10.490 -0500] MODIFY REQUEST conn=-3 op=190 msgID=191
origin="scim" dn="uid=user.3,ou=people,dc=example,dc=com"
```

SCIM monitoring

There are two facilities that can be used to monitor the SCIM activity in the server.

- **HTTPConnectionHandlerStatisticsMonitorProvider** -- Provides statistics straight about total and average active connections, requests per connection, connection duration, processing time, invocation count, etc.
- **SCIMServletMonitorProvider** -- Provides high level statistics about request methods (POST, PUT, GET, etc.), content types (JSON, XML), and response codes, for example, "`user-patch-404:26`".

The LDAP object class endpoints are treated as their own resource types, so that for requests using the Identity Access API, there will be statistics, such as `person-get-200` and `inetorgperson-post-401`.

Managing the SCIM 2.0 servlet extension

The PingDirectory Server provides a servlet extension that implements version 2.0 of the System for Cross-domain Identity Management (SCIM) specification. The SCIM 2.0 extension does not replace the existing SCIM 1.1 extension, and there are significant differences in how they are configured..

Supported SCIM 2.0 endpoints

PingDirectory Server supports the following SCIM 2.0 endpoints:

Endpoint	Description	Supported HTTP methods
/ServiceProviderConfig	Provides metadata that indicates the PingDirectory Server's authentication scheme (which is always OAuth 2.0) and its support for features that are optional in the SCIM standard. This endpoint is a metadata endpoint and is not subject to ACI restrictions.	`GET`
/Schemas	Lists the SCIM schemas that are configured for use on PingDirectory Server, and defines the various attributes available to resource types. This endpoint is a metadata endpoint and is not subject to ACI restrictions.	`GET`
/Schemas/<schema>	Retrieves a specific SCIM schema, as specified by the schema ID. This endpoint is a metadata endpoint and is not subject to ACI restrictions.	`GET`
/ResourceTypes	Lists all of the SCIM resource types that are configured for use on PingDirectory Server. Clients can use this information to determine the endpoint, core schema, and extension schemas of any resource types that the server supports. This endpoint is a metadata endpoint and is not subject to ACI restrictions.	`GET`
/ResourceTypes/<resourceType>	Retrieves a specific SCIM resource type, as specified by its ID. This endpoint is a metadata endpoint and is not subject to ACI restrictions.	`GET`
/<resourceType>	Creates a new resource (`POST`), or lists and filters resources (`GET`).	`GET`, `POST`

Endpoint	Description	Supported HTTP methods
/<resourceType>/.search	Lists and filters resources. This is used if passing query parameters in the URL is undesirable.	`POST`
/<resourceType>/<resourceId>	Retrieves a single resource (`GET`), modifies a single resource (`PUT`, `PATCH`), or deletes a single resource (`DELETE`).	`GET`, `PUT`, `PATCH`, `DELETE`

Configuring SCIM 2.0 on your server

Creating Your Own SCIM 2 application

The System for Cross-domain Identity Management (SCIM) is an open initiative designed to make moving identity data to, from, and between clouds standard, secure, fast, and easy. The SCIM SDK is a pre-packaged collection of libraries and extensible classes that provides developers with a simple, concrete API to interact with a SCIM service provider.

The SCIM 2 SDK is available for download at https://github.com/pingidentity/scim2.

Authentication requirements for SCIM 2.0 requests

All SCIM requests on a PingDirectory Server must use OAuth 2.0 bearer token authentication. Bearer tokens are evaluated using the SCIM 2 servlet extension's configured Access Token Validators. Basic authentication is not supported.

In addition to requiring bearer tokens, the server will not process any SCIM requests unless it has at least one encryption-settings definition in its encryption-settings database. You can create the necessary definition(s) with the encryption-settings command-line tool. See *Encrypting Sensitive Data* on page 544 for more information.

ⓘ **Note:** Encryption settings need to be defined on the Proxy and all backend PD servers, and the encryption settings should match.

Defining permissions for SCIM 2.0 requests

Whether or not a SCIM request is executed on a server depends primarily on both the configured Access Control Instructions (ACIs) in the server and the scopes which are present in the provided `OAuth` bearer token used to authenticate the request.

Internally, all SCIM 2.0 requests are processed using the cn=SCIM2 Servlet,cn=Root DNs,cn=config service account. Whether a requested operation is allowed depends on the ACIs that apply to the operation. The oauthscope bind rule is particularly useful for this, since it allows the administrator to use the supplied `OAuth` scopes in ACI logic.

Due to implementation details, access to the objectclass operational LDAP attribute is necessary for SCIM requests to properly execute. However, it is not advisable to give the service account access to objectclass on a global level. Instead, add the ACI granting objectclass access to the LDAP subtree you wish to expose to clients. See *Configuring permissions for SCIM 2.0 operations* on page 808 for an example of this.

ⓘ **Note:** ACIs that do not use the oauthscope bind rule can still apply to requested operations. For example, an ACI that grants unconditional read access to any authenticated LDAP user will also grant unconditional read access to SCIM requests regardless of the provided `OAuth` scopes, since the requests are processed through the service account.

SCIM 2.0 components

Unlike the SCIM 1.1 servlet extension, the SCIM 2.0 system is configured through the Administrative Console or with the **`dsconfig`** command-line tool. The SCIM 2.0 system consists of the following components:

- SCIM resource types
- SCIM schemas
  - SCIM attributes
  - SCIM sub-attributes
- SCIM attribute mappings (mapping resource types only)
- Correlated LDAP Data Views

A SCIM resource type defines a class of resources, such as users or devices. Every SCIM resource type features at least one SCIM schema, which defines the attributes that are available to each resource. If enabled for use, a SCIM resource type must also have a designated LDAP structural objectclass as well as an associated base DN.

The two types of SCIM resource types, mapping and passthrough, differ based on the definitions of the SCIM schema the resource types use.

- An LDAP mapping SCIM resource type requires an explicitly defined SCIM schema with explicitly defined mappings of SCIM attributes to LDAP attributes. Use a mapping SCIM resource type to exercise detailed control over the SCIM schema and its attributes and mappings.
- An LDAP passthrough SCIM resource type, by contrast, does not use an explicitly defined SCIM schema. Instead, an implicit schema is generated dynamically, based on the underlying LDAP schema. Use a passthrough SCIM resource type when you need to get started quickly.

A SCIM schema defines a collection of SCIM attributes, grouped under an identifier called a schema URN. Each SCIM resource type possesses a single core schema and can feature schema extensions, which act as secondary attribute groupings that the schema URN namespaces. SCIM Schemas are defined independently of SCIM resource types, and multiple SCIM resource types can use a single SCIM schema as a core schema or schema extension.

A SCIM attribute defines an attribute that is available under a SCIM schema. The configuration for a SCIM attribute defines its data type, regardless of whether it is required, single-valued, or multi-valued. When a SCIM attribute consists of SCIM sub-attributes, it is defined as a complex attribute.

A SCIM attribute mapping defines the manner in which a SCIM resource type maps the attributes in its SCIM schemas to native LDAP attributes of the PingDirectory Server.

A Correlated LDAP Data View allows a single SCIM resource that consists of attributes that are retrieved from multiple LDAP entries (see *Correlated LDAP data views* on page 803).

Correlated LDAP data views

Correlated LDAP data views can be attached to a SCIM resource type to allow that resource type to use attributes from other LDAP entries. A SCIM resource type can have multiple data views configured.

Correlated LDAP data views share many configuration settings with SCIM resource types. The following exceptions are below:

- primary-correlation-attribute: The LDAP attribute from the SCIM Resource Type whose value will be used to match entries in the data view. This property must be defined.
- secondary-correlation-attribute: The LDAP attribute from the data view whose value will be matched with the primary-correlation-attribute. This property must be defined.
- ldap-correlation-attribute-pair: Optional correlation attribute pairs. If these are configured, entries between the SCIM resource type and the correlated LDAP data view must also have matching values for the specified attributes for them to be returned together.

ⓘ **Note:** A a correlated LDAP Data View cannot provide attributes from more than one LDAP entry at a time. If there are multiple LDAP entries with secondary-correlation-attribute values that match the primary-correlation-attribute from the SCIM resource type's entry, an error will be thrown.

If the correlated LDAP data view is attached to a Passthrough SCIM resource type, its attributes will appear as schema extensions, similar to the following:

```
{
      ...
 "uid": [
     "user.8"
 ],
 "entryDN": "uid=user.8,ou=people,dc=example,dc=com",
 "urn:pingidentity:schemas:correlated:Document": {
         "documentIdentifier": [
             "user.8"
         ],
         "description": [
             "This is the description for the document user.8 under
 ou=Documents,dc=example,dc=com."
         ],
         ...
 },
 ...
 "schemas": [
     "urn:pingidentity:schemas:correlated:Document",
     "urn:pingidentity:schemas:passthrough:PassthroughUsers"
 ]
}
```

If the correlated LDAP data view is attached to a Mapping SCIM resource type, its attributes must be mapped to a schema used by the SCIM resource type. This is done through the correlated-ldap-data-view property in a SCIM attribute mapping.

Configuring an LDAP Mapping SCIM 2.0 resource type

About this task

This example shows how to add a simple mapping SCIM 2.0 resource type to a PingDirectory Server, backed by the inetOrgPerson LDAP objectclass. This example assumes that the PingDirectory Server has been configured using the default settings, meaning that sample data has been imported into the server and that data encryption has been set up.

Steps

1. Create the SCIM schema that the resource type will use:

```
dsconfig create-scim-schema \
--schema-name urn:pingidentity:schemas:User:1.0 \
--set display-name:User
```

2. Under this schema, add the following SCIM attributes.

```
dsconfig create-scim-attribute \
--schema-name urn:pingidentity:schemas:User:1.0 \
--attribute-name displayName
dsconfig create-scim-attribute \
--schema-name urn:pingidentity:schemas:User:1.0 \
--attribute-name name \
--set type:complex
dsconfig create-scim-subattribute \
--schema-name urn:pingidentity:schemas:User:1.0 \
--attribute-name name \
--subattribute-name familyName
dsconfig create-scim-subattribute \
--schema-name urn:pingidentity:schemas:User:1.0 \
--attribute-name name \
--subattribute-name formatted
dsconfig create-scim-attribute \
--schema-name urn:pingidentity:schemas:User:1.0 \
--attribute-name userName
```

3. Create the LDAP mapping SCIM resource type on the PingDirectory Server.

```
dsconfig create-scim-resource-type \
--type-name Users \
--type ldap-mapping \
--set enabled:true \
--set endpoint:Users \
--set structural-ldap-objectclass:inetOrgPerson \
--set include-base-dn:ou=People,dc=example,dc=com \
--set lookthrough-limit:500 \
--set core-schema:urn:pingidentity:schemas:User:1.0
```

4. Run the following commands to create the SCIM attribute mappings.

```
dsconfig create-scim-attribute-mapping \
--type-name Users \
--mapping-name displayName \
--set scim-resource-type-attribute:displayName \
--set ldap-attribute:displayName
dsconfig create-scim-attribute-mapping \
--type-name Users \
--mapping-name name.formatted \
--set scim-resource-type-attribute:name.formatted \
--set ldap-attribute:cn \
--set searchable:true
dsconfig create-scim-attribute-mapping \
--type-name Users \
--mapping-name name.familyName \
--set scim-resource-type-attribute:name.familyName \
--set ldap-attribute:sn \
--set searchable:true
dsconfig create-scim-attribute-mapping \
--type-name Users \
--mapping-name userName \
```

```
--set scim-resource-type-attribute:userName \
--set ldap-attribute:uid \
--set searchable:true
```

5. Configure the SCIM2 HTTP Servlet Extension to use a Mock Access Token Validator. Note that Mock Access Token Validators should never be used in production environments or with sensitive data.

```
dsconfig create-access-token-validator \
--validator-name "SCIM2 Mock Validator" \
--type mock \
--set enabled:true
dsconfig set-http-servlet-extension-prop \
--extension-name SCIM2 \
--set "access-token-validator:SCIM2 Mock Validator"
```

6. Send the following request to the SCIM /ResourceTypes endpoint to confirm that the new resource type has been added.

```
curl -k -X GET \
https://localhost:8443/scim/v2/ResourceTypes \
-H 'Authorization: Bearer {"active":true}'
```

7. The following JSON object should appear in the response in the “Resources” array:

```
{
...
"Resources": [{
"schemas":["urn:ietf:params:scim:schemas:core:2.0:ResourceType"],
"id":"Users",
"name":"Users",
"endpoint":"Users",
"schema":"urn:pingidentity:schemas:Users:1.0",
"meta":{
"resourceType":"ResourceType",
"location":"https://localhost:8443/scim/v2/ResourceTypes/Users"
}
}]
...
}
```

Configuring a correlated LDAP data view

About this task

The following example shows how to add a correlated LDAP data view to a LDAP Mapping SCIM Resource Type on a PingDirectory Server. The SCIM Resource Type will be a user, and the correlated LDAP data view will allow access to a document that matches their user ID.

For this examplel, we will use custom sample data, and then set up a new PingDirectory Server using this sample data.

Steps

1. Copy the following text and save it as `entries.ldif.template`.

```
define suffix=dc=example,dc=com
define maildomain=example.com
define numusers=101

branch: [suffix]
subordinateTemplate: admin:1
```

```
aci: (targetattr="*")(version 3.0; acl "Grant full access for the
 scim2allaccess OAuth 2 scope"; allow (all) oauthscope="scim2allaccess";)

branch: ou=People,[suffix]
subordinateTemplate: person:[numusers]

branch: ou=Documents,[suffix]
subordinateTemplate: document:[numusers]

template: admin
rdnAttr: uid
objectClass: top
objectClass: person
objectClass: organizationalPerson
objectClass: inetOrgPerson
uid: admin
givenName: Admin
sn: User
cn: Admin User
userPassword: password

template: person
rdnAttr: uid
objectClass: top
objectClass: person
objectClass: organizationalPerson
objectClass: inetOrgPerson
employeeNumber: <sequential:0>
uid: user.{employeeNumber}
sn: {uid}
cn: {uid}
userPassword: password

template: document
rdnAttr: documentIdentifier
objectClass: top
objectClass: document
documentIdentifier: user.<sequential:0>
description: This is the description for the document {documentIdentifier}
 under ou=Documents,dc=example,dc=com.
```

2. Run the following command:

```
$ bin/make-ldif --templateFile entries.ldif.template --ldifFile
 entries.ldif
```

3. Run setup for the PingDirectory Server. Make sure to import the created entries.ldif file, as well as set up encryption settings. After this is done, we will set up the SCIM resource type and the Correlated LDAP Data View.
4. Run the following command to define the SCIM schema:

```
"dsconfig create-scim-schema --schema-name urn:example:Users \
  --set "description:Users schema" --set display-name:Users
dsconfig create-scim-attribute --schema-name urn:example:Users \
  --attribute-name email --set required:true --set multi-valued:true
dsconfig create-scim-attribute --schema-name urn:example:Users \
  --attribute-name uid --set required:true --set mutability:read-only
dsconfig create-scim-attribute --schema-name urn:example:Users \
  --attribute-name documentId
dsconfig create-scim-attribute --schema-name urn:example:Users \
  --attribute-name documentDescription"
```

5. Run the following command to create the SCIM resource type:

```
dsconfig create-scim-resource-type \
  --type-name Users \
  --type ldap-mapping \
  --set core-schema:urn:example:Users \
  --set enabled:true \
  --set endpoint:Users \
  --set structural-ldap-objectclass:inetOrgPerson \
  --set include-base-dn:ou=people,dc=example,dc=com \
  --set create-dn-pattern:entryUUID=generated,ou=people,dc=example,dc=com
```

6. Run the following command to create the Correlated LDAP Data View:

```
dsconfig create-correlated-ldap-data-view \
  --type-name Users \
  --view-name Document \
  --set structural-ldap-objectclass:document \
  --set include-base-dn:ou=documents,dc=example,dc=com \
  --set create-dn-
pattern:entryUUID=generated,ou=documents,dc=example,dc=com \
  --set primary-correlation-attribute:uid \
  --set secondary-correlation-attribute:documentIdentifier
```

7. Run the following command to create the attribute mappings for the SCIM resource type attributes. Note that the correlated-ldap-data-view property is not set.

```
# The uid attribute, provided by the base SCIM Resource Type
dsconfig create-scim-attribute-mapping --type-name Users \
  --mapping-name uid \
  --set scim-resource-type-attribute:uid --set ldap-attribute:uid \
  --set writable:false --set searchable:true

# The email attribute, provided by the base SCIM Resource Type
dsconfig create-scim-attribute-mapping --type-name Users \
  --mapping-name email \
  --set scim-resource-type-attribute:email --set ldap-attribute:mail \
  --set searchable:true
```

8. Run the following command to create the DocumentId attribute mapping for the correlated LDAP data view attributes. The only real difference between mappings for SCIM resource type attributes and correlated LDAP data view attributes is the value of the correlated-ldap-data-view property.

```
# The documentId attribute
dsconfig create-scim-attribute-mapping --type-name Users \
  --mapping-name document.id \
  --set correlated-ldap-data-view:Document \
  --set scim-resource-type-attribute:documentId --set ldap-
attribute:documentIdentifier

# The documentDescription attribute
dsconfig create-scim-attribute-mapping --type-name Users \
  --mapping-name description \
  --set correlated-ldap-data-view:Document \
  --set scim-resource-type-attribute:documentDescription \
  --set ldap-attribute:description
```

9. This example uses a Mock Access Token Validator. This should not be done for production environments.

```
# Create a Mock Access Token Validator
dsconfig create-access-token-validator --validator-name "Mock ATV" \
```

```
  --type mock --set enabled:true --set evaluation-order-index:1000
# Configure SCIM 2 HTTP Servlet Extension to use Mock Access Token Val.
dsconfig set-http-servlet-extension-prop --extension-name SCIM2 \
  --set "access-token-validator:Mock ATV"
```

10. Run the following command to send a SCIM request

```
curl -k -X GET \
  https://localhost:8443/scim/v2/Users \
  -H 'Authorization: Bearer {"active":true, "scope":"scim2allaccess"}'
```

The response should look similar to the following. Notice that 'uid' and 'documentId' have the same value, since they are in a correlation attribute pair.

```
{
    "schemas": [
        "urn:ietf:params:scim:api:messages:2.0:ListResponse"
    ],
    "totalResults": 101,
    "Resources": [
        {
            "uid": "user.8",
            "id": "3715c022-1f34-36d9-bebc-7e74912106ec",
            "documentDescription": "This is the description \
            for the document user.8 under ou=Documents,dc=example,dc=com.,
            "documentId": "user.8",
            "meta": {
                "resourceType": "Users",
                "location": "https://localhost:8443/scim/v2/
Users/3715c022-1f34-36d9-bebc-7e74912106ec"
            },
            "schemas": [
                "urn:example:Users"
            ]
        },
    ...
}
```

Configuring permissions for SCIM 2.0 operations

About this task

This example shows how to correctly configure permissions so POST requests with the "userAdd" scope will succeed. This example assumes that you have set up an LDAP mapping SCIM 2.0 Resource Type for the inetOrgPerson objectclass (see *Configuring an LDAP Mapping SCIM 2.0 resource type* on page 803.

Steps

1. If the SCIM Resource Type being targeted already has a value for the create-dn-pattern property, skip to step 2. Otherwise, the following **`dsconfig`** command can be used:

```
dsconfig set-scim-resource-type-prop \
  --type-name Users \
  --set create-dn-pattern:entryUUID=generated,ou=People,dc=example,dc=com
```

2. Send the following request to the SCIM /Users endpoint:

```
curl -k -X POST \
  https://localhost:8443/scim/v2/Users/ \
  -H 'Authorization: Bearer {"active":true}' \
  -H 'Content-type: application/json' \
```

```
  --data '{"username":"user.test", "name":{"formatted":"Test",
"familyName":"User"}, "schemas":["urn:pingidentity:schemas:User:1.0"]}'
```

The response from the server should have a status of 403 and contain a correlation ID, similar to the following:

```
{
  "schemas":["urn:ietf:params:scim:api:messages:2.0:Error"],
  "status":"403",
  "detail":"Request failed:
correlationID='faa707b3-5d48-42e6-9e78-2c8dbb1e2cac'"
}
```

This is the expected response since this SCIM request does not have the permission needed to write to an entry. See *Troubleshoot the SCIM 2.0 servlet extension* on page 813 for instructions on viewing the full server error message.

3. An ACI should now be added to the ou=People,dc=example,dc=com subtree. Thist will grant permission to add entries to the said subtree as long as the SCIM request includes the ‘userAdd’ scope. This can be done by running the following **`ldapmodify`** command. Note that this ACI does not grant write access to attributes, which means modify operations will fail. See *Overview of access control* for more information on configuring ACIs.

```
$ ldapmodify
dn:ou=People,dc=example,dc=com
changetype:modify
add:aci
aci:(version 3.0; acl "ACI for userAdd scope"; allow (add)
oauthscope="userAdd";)
```

4. Send the POST request to the SCIM / Users endpoint again, this time including the userAdd scope in the bearer token:

```
curl -k -X POST \
https://localhost:8443/scim/v2/Users \
-H 'Authorization: Bearer {"active":true, "scope":"userAdd"}' \
-H 'Content-type: application/json' \
--data '{"username":"user.test", "name":{"formatted":"Test",
"familyName":"User"}, "schemas":["urn:pingidentity:schemas:User:1.0"]}'
```

The response from the server should now contain the created SCIM resource, which should also contain values for the name and username attributes, similar to the following:

```
{
"name":{
"familyName":"User",
"formatted":"Test"
},
"username":"user.test",
"id":"6f9a89b8-e766-478c-9667-def049daf6bc",
"meta":{
"resourceType":"Users",
"location":"https://localhost:8443/scim/v2/Users/6f9a89b8-e766-478c-9667-
def
049daf6bc"
},
"schemas":["urn:pingidentity:schemas:User:1.0"]
}
```

SCIM 2.0 searches

To prevent exhausting server resources, we recommend capping the total number of resources that are matched by a search. The configuration for each SCIM 2.0 resource type contains a lookthrough-limit property that defines this limit (the default lookthrough-limit value is 500).

If a search request exceeds the lookthrough limit, the client receives a 400 response with an error message similar to the following:

```
{
  "detail": "The search request matched too many results",
  "schemas": ["urn:ietf:params:scim:api:messages:2.0:Error"],
  "scimType": "tooMany",
  "status": "400"
}
```

To prevent this error, you have these options:

- The client must refine its search filter to return fewer matches.
- Configure paged searches as explained in *Using paged SCIM searches* on page 810.

Using paged SCIM searches

When searching large data sets, the results can be numerous and produce errors about a request matching too many results relative to the lookthrough limit. Paged searches avoid these errors and also reduce memory utilization.

About this task

PingDirectory does SCIM searches using LDAP requests. After you complete the steps below, PingDirectory creates LDAP requests that include request controls to sort and page the search results before returning the results.

If your SCIM searches result in an error because the request matched too many results, as discussed in *SCIM 2.0 searches* on page 810, you can avoid the error by using paged searches.

Complete the following steps for each search.

Steps

1. Decide your SCIM search.

> ⓘ **Note:** To get paged results, your search must include at least one of these parameters: startIndex, count, or sortBy.

For example, your search might look like the following search.

```
https://<directory-hostname>:<directory-port>/scim/v2/Users/?filter=st eq
 "TX"&sortBy=sn&sortOrder=ascending
```

Here is the corresponding encoded version.

```
https://<directory-hostname>:<directory-port>/scim/v2/Users/?filter=st
%20eq%20%22TX%22&sortBy=sn&sortOrder=ascending
```

On your PingDirectory Server, collect some information to use later. Find the SCIM resource type, `structural-ldap-objectclass`, `include-base-dn`, and `include-filter` values by running this command.

```
$ dsconfig get-scim-resource-type-prop --type-name <SCIM-resource-type-
name> \
--property structural-ldap-objectclass \
```

```
--property include-base-dn \
--property include-filter
```

2. On the PingDirectory server, complete the following steps.
   a. Create a Virtual List View (VLV) index for your search.

      Each SCIM search that you want to produce paged results must have its own VLV index.

      Create this index using `dsconfig create-local-db-vlv-index` with the following options.

Option	Description
`--index-name`	Names the index.
`--backend-name`	Specifies the name of the local database backend in which to place the index. The default database backend for PingDirectory is userRoot.
`--set base-dn`	Specifies the desired base dn. This value must match the value of the `include-base-dn` property that you found in the previous step.
`--set scope`	Is always `whole-subtree`.
`--set filter`	Specifies the filter. Specify "`(objectclass=`*`<name-of-SCIM-resource-type-objectclass>`*`)`" where *`<name-of-SCIM-resource-type-objectclass>`* is the name of the objectclass used by the SCIM resource type, which you found in the previous step. If the SCIM resource type has the `include-filter` property set, also specify that property value in the filter. For example, if the filter for the objectclass is `(objectclass=inetorgperson)` and the `include-filter` value is `(st=CA)`, specify the `--set filter` argument as "`(&(objectclass=inetorgperson)(st=CA))`". Specify the LDAP attributes for all the components of your SCIM search filter. For example, if a mapping SCIM resource type maps the LDAP attribute `st` to the SCIM attribute `address.region` and the SCIM search filter requires that `address.region eq TX`, then this filter must include `(st = TX)` instead of `(address.region = TX)`.

Option	Description
`--set sort-order`	Specifies whether to sort ascending (+) or descending (-) and the LDAP attribute to sort by. If the SCIM search does not specify the `sortBy` parameter, specify the sort order as `+entryUUID`.

Recall the original, decoded SCIM search, shown here.

```
https://<directory-hostname>:<directory-port>/scim/v2/Users/?filter=st
 eq "TX"&sortBy=sn&sortOrder=ascending
```

For example, to create a VLV index for that search, run the following command.

```
$ dsconfig create-local-db-vlv-index --index-name sn \
--backend-name userRoot --set base-dn:ou=people,dc=example,dc=com \
--set scope:whole-subtree \
--set filter:"(&(objectclass=inetorgperson)(st=TX))" --set sort-order:
+sn
```

b. Stop the server. Rebuild the index. Start the server. Run the `rebuild-index` command specifying the baseDN and the name of the index.

```
$ rebuild-index --baseDN <baseDN-value> --index <name-of-index>
```

For example, run these commands.

```
$ stop-server
$ rebuild-index --baseDN dc=example,dc=com --index vlv.sn
$ start-server
```

3. Run your SCIM search filter.

> ⓘ **Note:**
>
> The search can include only the filter you specified with `--set filter` in the earlier step without the "`(objectclass=<name-of-SCIM-resource-type-objectclass>)`" portion.

In addition to the Virtual List View request control, PingDirectory adds a Server Side request control to the LDAP request. These request controls require certain parameters be set. To satisfy this requirement, PingDirectory uses the following parameters. If the client does not provide values for one of the parameters, the search uses the corresponding default value shown in the following table.

Parameter	Default
startIndex	1
count	The value of the `lookthrough-limit` property of the SCIM resource type being searched. That default is 500.
sortBy	entryUUID With this default, the results appear unsorted.
sortOrder	ascending

SCIM 2.0 PATCH operations

When using a `PATCH` request to modify a SCIM 2.0 resource that has one or more required SCIM 2.0 attributes, the requester must have permissions to read the values of these required attributes in addition to write permissions for the attributes being modified, even if the `PATCH` request does not alter said requirements.

For example, assume we want to modify an LDAP Mapping SCIM 2.0 resource type using the following schema definition, where uid and cn are mapped to their LDAP equivalents:

```
{
  "schemas": ["urn:ietf:params:scim:schemas:core:2.0:Schema"],
  "id": "urn:test:schema:person",
  "attributes": [
    {
      "name": "uid",
      "type": "string",
      "multiValued": false,
      "required": true,
      "caseExact": false,
      "mutability": "readWrite",
      "returned": "default",
      "uniqueness": "none"
    },
    {
      "name": "cn",
      "type": "string",
      "multiValued": false,
      "required": false,
      "caseExact": false,
      "mutability": "readWrite",
      "returned": "default",
      "uniqueness": "none"
    }
  ],
  ...
        }
```

The following `PATCH` operation will fail if the SCIM 2 service account does not have access to both uid and cn:

```
{
  "schemas": ["urn:ietf:params:scim:api:messages:2.0:PatchOp"],
  "Operations":[{
        "op":"add",
        "path":"cn",
         "value": "new cn value”
  }]
        }
```

Troubleshoot the SCIM 2.0 servlet extension

For security reasons, error messages specifically regarding LDAP systems are suppressed and do not appear in the HTTP responses from the server. Instead, you will see something similar to the following:

```
{
  "schemas": [
    "urn:ietf:params:scim:api:messages:2.0:Error"
  ],
  "status": "400",
  "detail": "Request failed:
 correlationID='073eb1a8-8c51-48b3-83a0-380e1d4b4ab9'"
```

```
}
```

To view these messages, the Debug Trace Logger needs to be enabled. You can do this through the Administrative Console or with the following **dsconfig** command:

```
dsconfig set-log-publisher-prop --publisher-name "Debug Trace Logger" \
   --set enabled:true --add scim-message-type:error
```

After the Debug Trace Logger is enabled, the server will begin logging information related to SCIM operations to the file /logs/debug-trace , which will look somewhat like the following:

```
[09/Jun/2020:05:23:10.992 -0500] HTTP REQUEST requestID=3
correlationID="073eb1a8-8c51-48b3-83a0-380e1d4b4ab9" product="Ping Identity
Directory Server" instanceName="example" startupID="Xt9fJg==" threadID=173
from=[0:0:0:0:0:0:0:1]:53978 method=POST
url="https://0:0:0:0:0:0:0:1:9443/scim/v2/Users"
```

Note the presence of correlationID in these messages. By matching the id in the HTTP responses to the messages in the debug-trace log, the appropriate LDAP error message can be determined.

Disabling the SCIM 2.0 servlet extension

If you do not need to expose data through the SCIM 2 servlet, you can disable the SCIM 2.0 servlet extension by removing the SCIM2 HTTP servlet extension from the HTTPS connection handler, or from any other HTTP connection handler, and restart the handler:

```
dsconfig set-connection-handler-prop \
  --handler-name "HTTPS Connection Handler" \
  --remove http-servlet-extension:SCIM2 \
  --set enabled:false

dsconfig set-connection-handler-prop \
  --handler-name "HTTPS Connection Handler" \
  --set enabled:true
```

Managing Server SDK Extensions

The PingDirectory Server provides support for any custom extensions that you create using the Server SDK. This chapter summarizes the various features and extensions that can be developed using the Server SDK.

About the Server SDK

You can create extensions that use the Server SDK to extend the functionality of your Directory Server. Extension bundles are installed from a .zip archive or a file system directory. You can use the **manage-extension** tool to install or update any extension that is packaged using the extension bundle format. It opens and loads the extension bundle, confirms the correct extension to install, stops the server if necessary, copies the bundle to the server install root, and then restarts the server.

ⓘ **Note:** The **manage-extension** tool may only be used with Java extensions packaged using the extension bundle format. Groovy extensions do not use the extension bundle format. For more information, see the "Building and Deploying Java-Based Extensions" section of the Server SDK documentation, which describes the extension bundle format and how to build an extension.

Available types of extensions

The Server SDK provides support for creating a number of different types of extensions for Ping Identity Server Products, including the PingDirectory Server, PingDirectoryProxy Server, and PingDataSync Server. Some of those extensions include:

Cross-Product Extensions

- Access Loggers
- Alert Handlers
- Error Loggers
- Key Manager Providers
- Monitor Providers
- Trust Manager Providers
- OAuth Token Handlers
- Manage Extension Plugins

PingDirectory Server Extensions

- Certificate Mappers
- Change Subscription Handlers
- Extended Operation Handlers
- Identity Mappers
- Password Generators
- Password Storage Schemes
- Password Validators
- Plugins
- Tasks
- Virtual Attribute Providers

PingDirectoryProxy Server Extensions

- LDAP Health Checks
- Placement Algorithms
- Proxy Transformations

PingDataSync Server Extensions

- JDBC Sync Sources
- JDBC Sync Destinations
- LDAP Sync Source Plugins
- LDAP Sync Destination Plugins
- Sync SourcesSync Destinations
- Sync Pipe Plugins

For more information on the Server SDK, see the documentation available in the SDK build.

DevOps and infrastructure-as-code

Limitations when automating PingDirectory Server deployments

PingDirectory Server is a stateful application. User data is associated with it, and servers in a topology must be able to communicate bidirectionally with each other. The deployment of stateful applications is generally more challenging to automate than the deployment of stateless applications. However, by following certain industry-wide best practices, the deployment of stateful applications becomes easier to manage.

For stateful applications, we recommend maintaining a well-known network identifier for a server that does not change over its lifetime. Without this guarantee, the deployment automation workflows for the PingDirectory Server software does not work as expected. On infrastructure platforms like AWS, servers are generally assigned cattle-like internal host names. This strategy is acceptable if a well-defined external name is registered in a service discovery or lookup service, such as DNS.

Another important recommendation for stateful applications is the use of external, redundant persistent storage that is always available, and that functions independently of the server itself. Servers might come and go, but they are always guaranteed to be attached to the same persistent storage when they are resurrected. Although the PingDirectory Server software does not require this guarantee, we recommend it for simpler, less error-prone deployment automation and for easier disaster recovery. For more information, see *Deployment automation* on page 822.

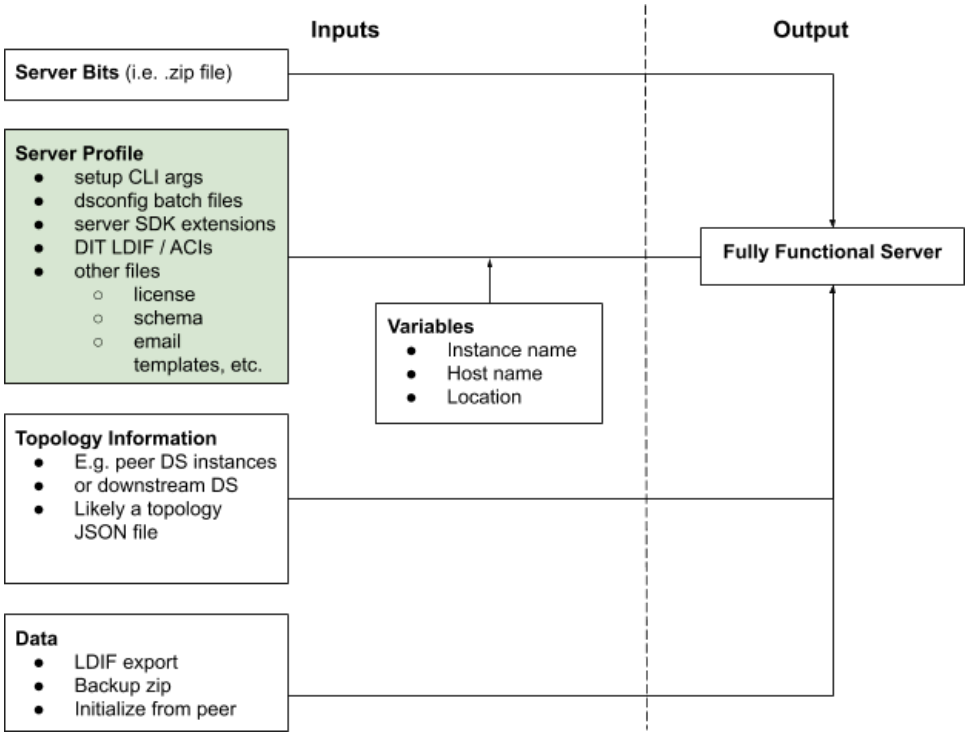
Server profiles

Regardless of the service model that your company follows, *server profiles* help you achieve your goals. At a basic level, a server profile defines a format for the configuration of a server by combining the following files into a single concrete structure:

- **`dsconfig`**
- Initial DIT
- Setup arguments
- Server SDK extensions
- Additional miscellaneous files

The primary goal of a server profile is to simplify the deployment of PingDirectory Server and related products by using deployment automation frameworks. When products support this capability, the amount of scripting that is required across automation frameworks – like Docker, Kubernetes, and Ansible – is reduced considerably.

The following image shows the role that a server profile plays in building a fully functional running server.



As a declarative form of a full server configuration, a server profile provides the following advantages:

- Provides a more complete and easily comparable way to define an individual server's configuration. Changes between different servers are easier to understand, and incremental changes to a server's configuration are easier to track.
- Ensures that each server instance is configured identically to its peers.
- Can be applied directly to new as well as previously installed instances.
- Shares a common configuration across a deployment pipeline of development, test, and production environments without unnecessary duplication. For information about substituting variables that differ by environment, see *Variable substitution* on page 817.
- Facilitates deployment automation by representing configuration as code.
- Reduces the number of additional configuration steps that are required to place a server into production.
- Makes the execution of various configuration changes more consistent and repeatable. The strategy of using a server profile to represent the final state of a server is less error-prone than recording a step-by-step process to attain that state.
- Can be managed easily in a version-control system.
- Simplifies the management of servers outside deployment automation frameworks.

A continuous deployment workflow can work with server profiles to make certain that changes to a server profile are moved automatically into production. In a stateful environment, the **`manage-profile replace-profile`** subcommand can be used to update existing servers. In a stateless environment, in which servers are considered immutable, **`manage-profile setup`** can be used to deploy new servers whenever a profile changes. With multiple environments, this deployment can be performed in a test environment before moving to production.

When working with zipped server SDK extensions and other files that might not be stored in a version-control system, the server profile can be modified to include these files prior to its use. For example, if the code for an extension is stored in a separate repository, it can be built and dropped into the server profile immediately before the **`manage-profile`** tool is run. This process is part of the deployment automation logic that uses the server profile, and it can be followed for any files that are needed by the server profile but whose versions are not controlled.

For more information about the **`manage-profile`** tool, see *About the manage-profile tool* on page 820.

Variable substitution

You can use the **`manage-profile`** tool to substitute different variables in server profiles.

The **`manage-profile`** tool uses the format **`${VARIABLE}`** to support the substitution of variables in profiles. This format can be escaped by using another **`$`**. For example, after substitution, **`$${VARIABLE}`** becomes **`${VARIABLE}`**.

Variable values can be read from a profile variables file or from environment variable values. If both options are used, the values that are specified in the file overwrite any environment variables.

The following code provides an example of how you can set user-defined variables by using a variables file in the server profile.

```
HOSTNAME=testserver.example.com
PORT=389
```

The following table describes built-in variables that can also be referenced in the server profile. Use these variables in the format previously described.

Built-in variable	Description
`PING_SERVER_ROOT`	Evaluates to the absolute path of the server's root directory
`PING_PROFILE_ROOT`	Evaluates to the individual profile's root directory

Built-in variable	Description
	ⓘ **Note:** Use PING_PROFILE_ROOT only with files that are not needed after initial setup, such as password files in `setup-arguments.txt`. Do not use the PING_PROFILE_ROOT variable for files needed while the server is running. The **`manage-profile`** tool creates a temporary copy of the server profile that is deleted after the tool completes, so files are not accessible under PING_PROFILE_ROOT when the server is running. For files you need while the server is running, such as keystore and truststore files, copy the files into the server root using the profile's `server-root/pre-setup` directory, and then refer to the files using with the PING_SERVER_ROOT variable.

For more information about the tool's usage, run the command **`bin/manage-profile --help`**.

Profile structure

Use either of the following methods to create a server profile:

- Use the template named `server-profile-template.zip`, which is located in the `resource/` directory.
- Run the **`manage-profile generate-profile`** subcommand. The **`manage-profile`** tool references a file system directory structure rather than a ZIP file.

Files can be added to each directory as needed.

The following hierarchy represents the file structure of a basic server profile:

```
-server-profile/
  |-- dsconfig/
  |-- ldif/
  |   |-- userRoot/
  |-- misc-files/
  |-- server-root/
  |   |-- post-setup/
  |   |-- pre-setup/
  |-- server-sdk-extensions/
  |-- setup-arguments.txt
  |-- variables-ignore.txt
```

setup-arguments.txt

When creating a profile, the first step is to add arguments to the file `setup-arguments.txt`. When **`manage-profile`** setup is run, these arguments are passed to the server's setup tool. To view the arguments that are available in this file, run the server's **`setup --help`** command.

To provide the equivalent, non-interactive CLI arguments after any prompts have been completed, run **`setup`** interactively. The `setup-arguments.txt` file in the profile template contains an example set of arguments that can be changed.

`setup-arguments.txt` is the only required file in the profile.

dsconfig/

dsconfig batch files can be added to the `dsconfig` directory. These files, each of which must include a `.dsconfig` extension, contain **dsconfig** commands to apply to server.

Because the **dsconfig** batch files are ordered lexicographically, `00-base.dsconfig` runs `before 01-second.dsconfig`, and so on.

To produce a `dsconfig` batch file that reproduces the current configuration, run **bin/config-diff**.

server-root/

Any server root files can be added to the `server-root` directory, including schema files, email template files, custom password dictionaries, and other files that must be present on the final server root. Add these files to the `server-root/pre-setup` or `server-root/post-setup` directory, depending on when they need to be copied to the server root. Most server root files are added to the server-root/pre-setup directory.

ldif/

Use LDIF files in the server profile to supply a base DIT, but not to import user data. Add LDIF files under the `ldif` directory. Place each LDIF file in a subdirectory that indicates the backend to which it is imported, such as `ldif/userRoot/` for the `userRoot` backend.

LDIF files require an `.ldif` extension and are ordered lexicographically.

server-sdk-extensions/

Add server SDK extension `.zip` files to the `server-sdk-extensions` directory. Include any configuration that is necessary for the extensions in the profile's **dsconfig** batch files.

variables-ignore.txt

The `variables-ignore.txt` file is an optional component of the server profile. It is useful when adding bash scripts to the server root because such files often contain expressions that the **manage-profile** tool normally interprets as variables.

Add `variables-ignore.txt` to a profile's root directory to indicate the relative paths of any files that are not to have their variables substituted.

The following example shows the contents of a typical `variables-ignore.txt` file:

```
server-root/pre-setup/script-to-ignore.sh
server-root/post-setup/another-file-to-ignore.txt
```

server-root/permissions.properties

The `permissions.properties` file, located in the `server-root` directory, is an optional file that specifies the permissions to apply to files that are copied to the server root. These permissions are represented in octal notation. By default, server root files maintain their permissions when copied.

The following example shows the contents of a typical `permissions.properties` file:

```
default=700
file-with-special-permissions.txt=600
new-subdirectory/file-with-special-permissions.txt=644
bin/example-script.sh=760
```

misc-files/

Additional documentation and other files can be added to the `misc-files` directory, which the **manage-profile** tool does not use. Use the variable **PING_PROFILE_ROOT** to refer to files in this directory from other locations, such as `setup-arguments.txt`.

> ⓘ **Note:**
>
> Use PING_PROFILE_ROOT only with files that are not needed after initial setup, such as password files in `setup-arguments.txt`. Do not use the PING_PROFILE_ROOT variable for files needed while the server is running. The **`manage-profile`** tool creates a temporary copy of the server profile that is deleted after the tool completes, so files are not accessible under PING_PROFILE_ROOT when the server is running. For files you need while the server is running, such as keystore and truststore files, copy the files into the server root using the profile's `server-root/pre-setup` directory, and then refer to the files using with the PING_SERVER_ROOT variable.

For example, a password file named `password.txt` in the `misc-files` directory could be referenced with `${PING_PROFILE_ROOT}/misc-files/password.txt` in `setup-arguments.txt`. Use a reference like this example to supply the file for the **`--rootUserPasswordFile`** argument in `setup-arguments.txt`.

About the manage-profile tool

The **`manage-profile`** tool is provided with the server to work with server profiles. It includes subcommands for creating, applying, and replacing server profiles, all of which significantly reduce the effort required by an administrator to configure a server appropriately.

The following sections describe these subcommands in more detail. For more information about the **`manage-profile`** tool, run **`manage-profile --help`**. For more information about each individual subcommand and its options, run **`manage-profile <subcommand> --help`**.

manage-profile generate-profile

To create a server profile from a configured server, use the **`generate-profile`** subcommand. The generated profile contains the following information, which provides a base for completing a profile:

- Command-line arguments that were used to set up the server
- **`dsconfig`** commands necessary to configure the server
- Installed server SDK extensions
- Files that are added to the server root

To produce a complete profile, some parts of the generated profile might require modifications, such as adding password files that `setup-arguments.txt` uses. The **`--instanceName`** and **`--localHostName`** arguments in `setup-arguments.txt` are made variables by **`generate-profile`**, and must be provided values when other **`manage-profile`** subcommands use the generated profile.

LDIF files must also be added manually to the generated profile.

manage-profile setup

To apply a server profile to a fresh, unconfigured server, use the **`setup`** subcommand, which replaces the normal setup tool when using a server profile. Run **`manage-profile setup`** to complete the following tasks:

- Copies the pre-setup files to the server root
- Runs the setup tool
- Copies the post-setup files to the server root
- Installs any server SDK extensions
- Runs any dsconfig batch files
- Imports any LDIF files
- Installs the server SDK extensions
- Starts the server

While manage-profile setup is running, a copy of the profile is created in a temporary directory that can be specified by using the **`--tempProfileDirectory`** argument. The command leaves the server in a complete and running state when finished, unless the **`--doNotStart`** argument is specified.

manage-profile replace-profile

Run the **replace-profile** subcommand on a server that was originally set up with a server profile to replace its configuration with a new profile. The tool applies a specified server profile to an existing server while preserving its data, topology configuration, and replication configuration. New LDIF files from the replacement server profile are not imported.

While **manage-profile replace-profile** is running, the existing server is stopped and moved to a temporary directory that the **--tempServerDirectory** argument can specify. A fresh, new server is subsequently installed and set up with the new profile. The final server is left running if it was running before the command was started, and remains stopped if it was stopped.

Run **manage-profile replace-profile** from a second unzipped server install package on the same host as the existing server, similar to the **update** tool. Use the **--serverRoot** argument to specify the root of the existing server that will have its profile replaced.

If files have been added or modified in the server root since the most recent **manage-profile setup** or **manage-profile replace-profile** was run, they are included in the final server with the replaced profile. Otherwise, files specifically added from the `server-root` directory of the previous server profile are absent from the final server with the replaced profile. If errors occur during the subcommand, such as the new profile having an invalid `setup-arguments.txt` file, the existing server returns to its original state from before **manage-profile replace-profile** was run.

> ⓘ **Note:** The **manage-profile replace-profile** tool can update the server version when needed.

> ⓘ **Note:** The **manage-profile replace-profile** tool can directly apply configuration changes when there are no other changes in the new profile. This is a shorter process when making small changes to **dsconfig**.

Server Profiles in a Pets Service Model

Server profiles and other DevOps concepts are also invaluable in a pets service model. For example, the step of using the **manage-profile generate-profile** subcommand to generate a server profile from a production server creates an easily consumable representation of the server's configuration. In nearly every scenario, the generation of a profile from an existing server is simpler than the piecing together manually of schemas, extensions, and other configuration information to create an image of that server. Additionally, generated profiles can be backed up or checked in to source control to maintain a consistent picture of an active server's configuration.

Another valuable use of server profiles involves setting up servers in a test environment that is separate from production. For example, a profile that matches the profile of a production server can be generated and used to install a fresh test server that matches the production server. Further, variable substitution allows environmental changes, such as local host name or instance name, without requiring a separate profile. Because the server's original configuration matches the running production server, adjustments can be tested easily. This approach provides more consistency when you validate changes before moving them to production.

If a new pet server has been set up with a server profile, **manage-profile replace-profile** can be used to apply changes to the profile. Rather than using scripts or a manual process to apply individual changes, **replace-profile** provides a consistent, repeatable method of moving to a new server profile. This strategy automates more easily and is less prone to human error.

For more information about the **manage-profile** tool, see *About the manage-profile tool* on page 820.

Topology-management tools

Because PingDirectory Server topology-management tools like **dsreplication** and **remove-defunct-server** feature internal retries, external deployment automation scripts are not required to

incorporate retries. This approach mitigates transient failures and makes servers more robust in automated environments.

To support this approach, **dsreplication** and **remove-defunct-server** include a **--retryTimeoutSeconds** option, which specifies a timeout value for the entire command. If the command fails, it is retried until the timeout value is reached. The command is not aborted if the timeout expires mid-execution, so it is guaranteed to be executed at least once. The default value of zero indicates the command does not have a timeout and is not retried upon initial failure.

The **remove-defunct-server tool** can also be used to complete the following tasks:

- Remove an online server.

  Because a server runs as a container's main process, and because shutting down a server re-spins the container, this task was impossible to complete in deployment automation environments that use containers. Automation scripts often resorted to hacks like impersonating an offline server by taking down the LDAP connection handler. In environments where security concerns prevented the use of LDAP, removing an online server was never an option.
- Remove servers without forcing a topology master when you do not have a quorum majority of online severs.

  The topology registry uses a master-slave architecture in which all write requests are chained through a single write-master. When a master could not be nominated, the topology becomes read-only. To work around this issue, a single server had to be forced as a master to apply changes.
- Remove servers concurrently.

The **--ignoreOnline** option can be used to remove an online server, and the **--retryTimeoutSeconds** option can be used to increase robustness in concurrent environments.

Finally, **dsreplication enable** provides internal support for a seed server before the replication topology exists. To prevent the creation of separate islands of replicating servers when simultaneously enabling replication on multiple hosts with a topology JSON file, the server that appears first in lexicographical order in the topology JSON file is designated as the seed server. When enabling replication on the seed server itself with the topology JSON file, it stops and returns a `ERROR_SEED_AND_TARGET_SERVER_ARE_THE_SAME` code. This precaution avoids deadlocking a **dsreplication** process when replication is invoked concurrently. On the seed server itself, deployment automation scripts can treat this code as being successful.

Deployment automation

Automated workflows help shift the deployment process from a pets service model to a cattle service model. The primary tools that are required to manage the replication topology are **dsreplication** and **remove-defunct-server**.

Another key to every topology-management workflow is the `topology.json` file, which represents the intended state of the topology at any time. An administrator should create this file manually or by using external automation. When promoting from a dev/test environment to a stage/prod environment, you can obtain a `topology.json` file using the **manage-topology export** subcommand.

The following code shows an example `topology.json` file:

```
{
  "serverInstances" : [
    {
      "instanceName" : "ds-0",
      "hostname" : "ds-0.ds-topology.production.svc.cluster.local",
      "location" : "Austin",
      "ldapPort" : 389,
      "ldapsPort" : 636,
      "replicationPort" : 989,
      "startTLSEnabled" : true,
      "preferredSecurity" : "SSL",
```

```
      "product" : "DIRECTORY"
    },
    {
      "instanceName" : "ds-1",
      "hostname" : "ds-1.ds-topology.production.svc.cluster.local",
      "location" : "Austin",
      "ldapPort" : 389,
      "ldapsPort" : 636,
      "replicationPort" : 989,
      "startTLSEnabled" : true,
      "preferredSecurity" : "SSL",
      "product" : "DIRECTORY"
    },
    {
      "instanceName" : "ds-2",
      "hostname" : "ds-2.ds-topology.production.svc.cluster.local",
      "location" : "Austin",
      "ldapPort" : 389,
      "ldapsPort" : 636,
      "replicationPort" : 989,
      "startTLSEnabled" : true,
      "preferredSecurity" : "SSL",
      "product" : "DIRECTORY"
    },
    {
      "instanceName" : "ds-3",
      "hostname" : "ds-3.ds-topology.production.svc.cluster.local",
      "location" : "Austin",
      "ldapPort" : 389,
      "ldapsPort" : 636,
      "replicationPort" : 989,
      "startTLSEnabled" : true,
      "preferredSecurity" : "SSL",
      "product" : "DIRECTORY"
    },
    ...
  ]
}
```

The remaining sections in this chapter describe the deployment automation that is necessary to satisfy the following workflows:

- Setting up the initial topology
- Initializing data on all servers
- Replacing crashed instances and scaling up
- Scaling down
- Rolling updates

Consistent network identifiers are required for each server instance. Additionally, we strongly recommend persistent storage for all server bits. The required level of automation changes slightly when this recommendation is not followed, as noted in each section.

Setting up the initial topology

About this task

The `server.uuid` file in the server's `config` directory indicates whether the server is being set up initially or being updated.

ⓘ **Note:**

If you're not using persistent storage, set up a new server and skip step 3. If you are using persistent storage, start on step 3.

For more information about server profiles, see *Server profiles* on page 816.

Steps

1. Create or obtain the `topology.json` file either manually or by using external automation.

   When promoting from a dev/test environment to a stage/prod environment, you can obtain a `topology.json` file using `manage-topology export`.
2. To install and configure a server instance, run the **`manage-profile`** command with the `setup` option.

```
manage-profile setup \
  --profile /path/to/server-profile \
  --profileVariablesFile /path/to/instance-specific-variables.properties
```

3. If you're using persistent storage, replace the current server profile. Otherwise, proceed to step 4. To replace the current profile, run the `manage-profile replace-profile` command from a second, extracted server install package on the same host as the existing server.

```
manage-profile replace-profile \
  --profile /path/to/server-profile \
  --profileVariablesFile /path/to/instance-specific-variables.properties \
  --serverRoot /path/to/existing/server
```

4. To enable replication within the topology, run **`dsreplication`** with the `enable` option.

```
dsreplication enable \
  --topologyFilePath /path/to/topology.json \
  --retryTimeoutSeconds 120
```

5. To initialize the replication data, run **`dsreplication`** with the `initialize` option.

```
dsreplication initialize \
  --topologyFilePath /path/to/topology.json
```

Initializing data on all servers

About this task

Skip this step if you are using one of the following methods to load data during the initial setup:

- Each server instance imports the same user data from one or more LDIF backup files
- Data is synchronized from another source, such as an Active Directory server, to all servers
- No data is imported initially

Data must be imported and available on a single server only. Servers that do not feature user data must be initialized reliably and quickly.

ⓘ **Note:** The absence of user data during the initial setup does not prevent the the environment from being set up correctly.

Steps

1. Create or obtain the `topology.json` file either manually or by using external automation. When promoting from a dev/test environment to a stage/prod environment, you can obtain a `topology.json` file using the **`manage-topology export`** subcommand.

2. Import data into one of the server instances, as follows:

```
import-ldif --backendID userRoot \
  --ldifFile userData0.ldif \
  --ldifFile userData1.ldif
```

3. Initialize data on each server by using this server as the initial source:

```
dsreplication initialize \
  --topologyFilePath /path/to/topology.json \
  --retryTimeoutSeconds 120
```

For the sake of simplicity, other **`dsreplication`** options are not shown.

Replacing crashed instances and scaling up

About this task

The automation for this scenario is identical to the automation for *Setting up the initial topology* on page 823.

Scaling down

About this task

Steps

1. Create or obtain the `topology.json` file either manually or by using external automation. When promoting from a dev/test environment to a stage/prod environment, you can obtain a `topology.json` file using the **`manage-topology export`** subcommand.
2. If the server is not present in the topology JSON file, remove it from the topology, as follows:

```
remove-defunct-server --topologyFilePath /path/to/topology.json \
  --ignore-online \
  --serverInstanceName instance-name
  --retryTimeoutSeconds 120
```

For the sake of simplicity, LDAP bind options are omitted.

Rolling updates

About this task

The presence of the file `server.uuid` in the server's `config` directory indicates whether the server is being set up initially or being updated.

Steps

1. Create or obtain the `topology.json` file either manually or by using external automation. When promoting from a dev/test environment to a stage/prod environment, you can obtain a `topology.json` file using the **`manage-topology export`** subcommand.
2. Unzip the new server bits to a directory.
3. From the same directory's `bin` directory, replace the new server profile, as follows:

```
manage-profile replace-profile \
  --profile /path/to/server-profile \
  --profileVariablesFile /path/to/instance-specific-variables.properties \
  --serverRoot /path/to/existing/server
```

Troubleshooting the Server

The PingDirectory Server provides a highly-reliable service that satisfies your company's objectives. However, if problems do arise (whether from issues in the Directory Server itself or a supporting component, like the JVM, operating system, or hardware), then it is essential to be able to diagnose the problem quickly to determine the underlying cause and the best course of action to take towards resolving it.

This chapter provides information about how to perform this analysis to help ensure that the problem is resolved as quickly as possible. This chapter presents the following information:

Working with the collect-support-data tool

The Directory Proxy Server provides a significant amount of information about its current state including any problems that it has encountered during processing. If a problem occurs, the first step is to run the **collect-support-data** tool in the `bin` directory. The tool aggregates all relevant support files into a zip file that administrators can send to your authorized support provider for analysis. The tool also runs data collector utilities, such as **jps**, **jstack**, and **jstat** plus other diagnostic tools, and bundles the results in the zip file.

The tool may only archive portions of certain log files to conserve space, so that the resulting support archive does not exceed the typical size limits associated with e-mail attachments.

The data collected by the **collect-support-data** tool varies between systems. However, the tool always tries to get the same information across all systems for the target Directory Proxy Server. The data collected includes the configuration directory, summaries and snippets from the `logs` directory, an LDIF of the monitor and RootDSE entries, and a list of all files in the server root.

Server commands used in the collect-support-data tool

The following presents a summary of the data collectors that the **collect-support-data** tool archives in zip format. If an error occurs during processing, you can re-run the specific data collector command and send the results to your authorized support provider.

Directory Server Commands Used in the Collect-Support-Data Tool

Data Collector	Description
status	Runs **status** `-F` to show the full version information of the Directory Server (Unix, Windows).
server-state	Runs **server-state** to show the current state of the Directory Server process (Unix, Windows).

JDK commands used in the collect-support-data tool

JDK Commands Used in the Collect-Support-Data Tool

Data Collector	Description
jps	Java Virtual Machine Process status tool. Reports information on the JVM (Linux, Windows, Mac OS).
jstack	Java Virtual Machine Stack Trace. Prints the stack traces of threads for the Java process (Linux, Windows, Mac OS).
jstat	Java Virtual Machine Statistics Monitoring Tool. Displays performance statistics for the JVM (Linux, Windows, Mac OS).

Data Collector	Description
jinfo	Displays the Java configuration information for the Java process (Linux, Windows, Mac OS).

Linux commands used in the collect-support-data tool

Linux Commands Used in the Collect-Support-Data Tool

Data Collector	Description
tail	Displays the last few lines of a file. Tails the `/var/logs/messages` directory.
uname	Prints system, machine, and operating system information.
ps	Prints a snapshot of the current active processes.
df	Prints the amount of available disk space for file systems in 1024-byte units.
cat	Concatenates the following files and prints to standard output: ▪ /proc/cpuinfo ▪ /proc/meminfo ▪ /etc/hosts ▪ /etc/nsswitch.conf ▪ /etc/resolv.conf
netstat	Prints the state of network interfaces, protocols, and the kernel routing table.
ifconfig	Prints information on all interfaces.
uptime	Prints the time the server has been up and active.
dmesg	Prints the message buffer of the kernel.
vmstat	Prints information about virtual memory statistics.
iostat	Prints disk I/O and CPU utilization information.
mpstat	Prints performance statistics for all logical processors.
pstack	Prints an execution stack trace on an active processed specified by the pid.
top	Prints a list of active processes and how much CPU and memory each process is using.

MacOS commands used in the collect-support-data tool

MacOS Commands Used in the Collect-Support-Data Tool

Data Collector	Description
uname	Prints system, machine, and operating system information.
uptime	Prints the time the server has been up and active.
ps	Prints a snapshot of the current active processes.
system_profiler	Prints system hardware and software configuration.
vm_stat	Prints machine virtual memory statistics.
tail	Displays the last few lines of a file. Tails the `/var/log/system.log` directory.
netstat	Prints the state of network interfaces, protocols, and the kernel routing table.

Data Collector	Description
ifconfig	Prints information on all interfaces.
df	Prints the amount of available disk space for file systems in 1024-byte units.
sample	Profiles a process during an interval.

Invoking the collect-support-data tool as an administrative task

You can invoke **`collect-support-data`** as an administrative task, including as a recurring task that can be automatically invoked on a regular basis.

For more information, see *About recurring tasks and task chains* on page 308.

Available tool options

The **`collect-support-data`** tool has some important options that you should be aware of:

- **--noLdap**. Specifies that no effort should be made to collect any information over LDAP. This option should only be used if the server is completely unresponsive or will not start and only as a last resort.
- **--pid {pid}**. Specifies the ID of an additional process from which information is to be collected. This option is useful for troubleshooting external server tools and can be specified multiple times for each external server, respectively.
- **--sequential**. Use this option to diagnose “Out of Memory” errors. The tool collects data in parallel to minimize the collection time necessary for some analysis utilities. This option specifies that data collection should be run sequentially as opposed to in parallel. This action has the effect of reducing the initial memory footprint of this tool at a cost of taking longer to complete.
- **--reportCount {count}**. Specifies the number of reports generated for commands that supports sampling (for example, **`vmstat`**, **`iostat`**, or **`mpstat`**). A value of 0 (zero) indicates that no reports will be generated for these commands. If this option is not specified, it defaults to 10.
- **--reportInterval {interval}**. Specifies the number of seconds between reports for commands that support sampling (for example, **`mpstat`**). This option must have a value greater than 0 (zero). If this option is not specified, it default to 1.
- **--maxJstacks {number}**. Specifies the number of jstack samples to collect. If not specified, the default number of samples collected is 10.
- **--collectExpensiveData**. Specifies that data on expensive or long running processes be collected. These processes are not collected by default, because they will impact the performance of a running server.
- **--comment {comment}**. Provides the ability to submit any additional information about the collected data set. The comment will be added to the generated archive as a README file.
- **--includeBinaryFiles**. Specifies that binary files be included in the archive collection. By default, all binary files are automatically excluded in data collection.
- **--adminPassword {adminPassword}**. Specifies the global administrator password used to obtain **`dsreplication status`** information.
- **--adminPasswordFile {adminPasswordFile}**. Specifies the file containing the password of the global administrator used to obtain **`dsreplication status`** information.
- **--outputPath**. Specifies the path (and optionally, the name) for the support data archive file. If the path specifies a filename, the archive is written to that file. If the path specifies a directory, the file is written into that directory with a server-generated name.
- **--useRemoteServer**. Invokes the tool against a remote server instance and streams the output and resulting support data archive back to the client. This option can be especially useful when the server instance is running in a container because it might otherwise be difficult to invoke commands or access files in that container. See also: **--proxyToServerAddress** and **--proxyToServerPort**.
- **--proxyToServerAddress** and **--proxyToServerPort**. Use these options with **--useRemoteServer** to indicate that the support data archive should be retrieved from a server that is not directly accessible but can be accessed through a Directory Proxy Server.

Running the collect-support-data tool

Steps

1. Go to the server root directory.
2. Use the **collect-support-data** tool. Make sure to include the host, port number, bind DN, and bind password.

```
$ bin/collect-support-data --hostname 127.0.0.1 --port 389 \
  --bindDN "cn=Directory Manager" --bindPassword secret \
  --serverRoot /opt/PingDirectory --pid 1234
```

3. Email the zip file to your Authorized Support Provider.

Directory Server Troubleshooting information

The Directory Server has a comprehensive default set of log files and monitor entries that are useful when troubleshooting a particular server problem.

Error log

By default, this log file is available at `logs/errors` below the server install root and it provides information about warnings, errors, and other significant events that occur within the server. A number of messages are written to this file on startup and shutdown, but while the server is running there is normally little information written to it. In the event that a problem does occur, however, the server writes information about that problem to this file.

The following is an example of a message that might be written to the error log:

```
[11/Apr/2011:10:31:53.783 -0500] category=CORE severity=NOTICE msgID=458887
 msg="The Directory Server has started successfully"
```

The category field provides information about the area of the server from which the message was generated. Available categories include:

ACCESS_CONTROL, ADMIN, ADMIN_TOOL, BACKEND, CONFIG, CORE, DSCONFIG, EXTENSIONS, PROTOCOL, SCHEMA, JEB, SYNC, LOG, PLUGIN, PROXY, QUICKSETUP, REPLICATION, RUNTIME_INFORMATION, TASK, THIRD_PARTY, TOOLS, USER_DEFINED, UTIL, VERSION.

The severity field provides information about how severe the server considers the problem to be. Available severities include:

- **DEBUG** – Used for messages that provide verbose debugging information and do not indicate any kind of problem. Note that this severity level is rarely used for error logging, as the Directory Server provides a separate debug logging facility as described below.
- **INFORMATION** – Used for informational messages that can be useful from time to time but are not normally something that administrators need to see.
- **MILD_WARNING** – Used for problems that the server detects, which can indicate something unusual occurred, but the warning does not prevent the server from completing the task it was working on. These warnings are not normally something that should be of concern to administrators.
- **MILD_ERROR** – Used for problems detected by the server that prevented it from completing some processing normally but that are not considered to be a significant problem requiring administrative action.
- **NOTICE** – Used for information messages about significant events that occur within the server and are considered important enough to warrant making available to administrators under normal conditions.
- **SEVERE_WARNING** – Used for problems that the server detects that might lead to bigger problems in the future and should be addressed by administrators.
- **SEVERE_ERROR** – Used for significant problems that have prevented the server from successfully completing processing and are considered important.

- **FATAL_ERROR** – Used for critical problems that arise which might leave the server unable to continue processing operations normally.

The messages written to the error log may be filtered based on their severities in two ways. First, the error log publisher has a `default-severity` property, which may be used to specify the severity of messages logged regardless of their category. By default, this includes the NOTICE, SEVERE_WARNING, SEVERE_ERROR, and FATAL_ERROR severities.

You can override these severities on a per-category basis using the `override-severity` property. If this property is used, then each value should consist of a category name followed by an equal sign and a comma-delimited set of severities that should be logged for messages in that category. For example, the following override severity would enable logging at all severity levels in the PROTOCOL category:

```
protocol=debug,information,mild-warning,mild-error,notice,severe-
warning,severe-error,fatal-error
```

Note that for the purposes of this configuration property, any underscores in category or severity names should be replaced with dashes. Also, severities are not inherently hierarchical, so enabling the DEBUG severity for a category will not automatically enable logging at the INFORMATION, MILD_WARNING, or MILD_ERROR severities.

The error log configuration may be altered on the fly using tools like **`dsconfig`**, the Administrative Console, or the LDIF connection handler, and changes will take effect immediately. You can configure multiple error logs that are active in the server at the same time, writing to different log files with different configurations. For example, a new error logger may be activated with a different set of default severities to debug a short-term problem, and then that logger may be removed once the problem is resolved, so that the normal error log does not contain any of the more verbose information.

server.out log

The `server.out` file holds any information written to standard output or standard error while the server is running. Normally, it includes a number of messages written at startup and shutdown, as well as information about any administrative alerts generated while the server is running. In most cases, this information is also written to the error log. The `server.out` file can also contain output generated by the JVM. For example, if garbage collection debugging is enabled, or if a stack trace is requested via "kill -QUIT" as described in a later section, then output is written to this file.

Debug log

The debug log provides a means of obtaining information that can be used for troubleshooting problems but is not necessary or desirable to have available while the server is functioning normally. As a result, the debug log is disabled by default, but it can be enabled and configured at any time.

Some of the most notable configuration properties for the debug log publisher include:

- **enabled** – Indicates whether debug logging is enabled. By default, it is disabled.
- **log-file** – Specifies the path to the file to be written. By default, debug messages are written to the `logs/debug` file.
- **default-debug-level** – Specifies the minimum log level for debug messages that should be written. The default value is “error,” which only provides information about errors that occur during processing (for example, exception stack traces). Other supported debug levels include warning, info, and verbose. Note that unlike error log severities, the debug log levels are hierarchical. Configuring a specified debug level enables any debugging at any higher levels. For example, configuring the info debug level automatically enables the warning and error levels.
- **default-debug-category** – Specifies the categories for debug messages that should be written. Some of the most useful categories include caught (provides information and stack traces for any exceptions caught during processing), database-access (provides information about operations performed in the underlying database), protocol (provides information about ASN.1 and LDAP communication performed by the server), and data (provides information about raw data read from or written to clients).

As with the error and access logs, multiple debug loggers can be active in the server at any time with different configurations and log files to help isolate information that might be relevant to a particular problem.

ⓘ **Note:** Enabling one or more debug loggers can have a significant impact on server performance. We recommend that debug loggers be enabled only when necessary, and then be scoped so that only pertinent debug information is recorded.

Debug targets can be used to further pare down the set of messages generated. For example, you can specify that the debug logs be generated only within a specific class or package. If you need to enable the debug logger, you should work with your authorized support provider to best configure the debug target and interpret the output.

Replication repair log

The replication repair log is written to `logs/replication` by default and records information about processing performed by the replication repair service. This log is used to resolve replication conflicts that can arise. For example, if the same entry is modified at the same time on two different systems, or if an attempt is made to create entries with the same DN at the same time on two different systems, the Directory Server records these events.

Config audit log and the configuration archive

The configuration audit log provides a record of any changes made to the server configuration while the server is online. This information is written to the `logs/config-audit.log` file and provides information about the configuration change in the form that may be used to perform the operation in a non-interactive manner with the **`dsconfig`** command. Other information written for each change includes:

- Time that the configuration change was made.
- Connection ID and operation ID for the corresponding change, which can be used to correlate it with information in the access log.
- DN of the user requesting the configuration change and the method by which that user authenticated to the server.
- Source and destination addresses of the client connection.
- Command that can be used to undo the change and revert to the previous configuration for the associated configuration object.

In addition to information about the individual changes that are made to the configuration, the Directory Server maintains complete copies of all previous configurations. These configurations are provided in the `config/archived-configs` directory and are gzip-compressed copies of the `config/config.ldif` file in use before the configuration change was made. The file names contain timestamps that indicate when that configuration was first used.

Access and audit log

The access log provides information about operations processed within the server. The default access log file is written to `logs/access`, but multiple access loggers can be active at the same time, each writing to different log files and using different configurations.

By default, a single access log message is generated, which combines the elements of request, forward, and result messages. If an error is encountered while attempting to process the request, then one or more forward-failed messages may also be generated.

```
[01/Jun/2011:11:10:19.692 -0500] CONNECT conn=49 from="127.0.0.1"
 to="127.0.0.1"
  protocol="LDAP+TLS" clientConnectionPolicy="default"
[01/Jun/2011:11:10:19.764 -0500] BIND RESULT conn=49 op=0 msgID=1 version="3"
  dn="cn=Directory Manager" authType="SIMPLE" resultCode=0 etime=0.401
  authDN="cn=Directory Manager,cn=Root DNs,cn=config"
 clientConnectionPolicy="default"
```

```
[01/Jun/2011:11:10:19.769 -0500] SEARCH RESULT conn=49 op=1 msgID=2
  base="ou=People,dc=example,dc=com" scope=2 filter="(uid=1)" attrs="ALL"
  resultCode=0 etime=0.549 entriesReturned=1
[01/Jun/2011:11:10:19.788 -0500] DISCONNECT conn=49 reason="Client Unbind"
```

Each log message includes a timestamp indicating when it was written, followed by the operation type, the connection ID (which is used for all operations processed on the same client connection), the operation ID (which can be used to correlate the request and response log messages for the operation), and the message ID used in LDAP messages for this operation.

The remaining content for access log messages varies based on the type of operation being processed, and whether it is a request or a result message. Request messages generally include the most pertinent information from the request, but generally omit information that is sensitive or not useful.

Result messages include a `resultCode` element that indicates whether the operation was successful or if failed and an `etime` element that indicates the length of time in milliseconds that the server spent processing the operation. Other elements that might be present include the following:

- **origin=replication** – Operation that was processed as a result of data synchronization (for example, replication) rather than a request received directly from a client.
- **message** – Text that was included in the `diagnosticMessage` field of the response sent to the client.
- **additionalInfo** – Additional information about the operation that was not included in the response sent back to the client.
- **authDN – DN** of the user that authenticated to the server (typically only included in bind result messages).
- **authzDN – DN** of an alternate authorization identify used when processing the operation (for example, if the proxied authorization control was included in the request).
- **authFailureID** – Unique identifier associated with the authentication failure reason (only included in non-successful bind result messages).
- **authFailureReason** – Information about the reason that a bind operation failed that might be useful to administrators but was not included in the response to the client for security reasons.
- **responseOID** – OID included in an extended response returned to the client.
- **entriesReturned** – Number of matching entries returned to the client for a search operation.
- **unindexed=true** – Indicates that the associated search operation could not be sufficiently processed using server indexes and a significant traversal through the database was required.

Note that this is not an exhaustive list, and elements that are not listed here may also be present in access log messages. The LDAP SDK for Java provides an API for parsing access log messages and provides access to all elements that they may contain.

The Directory Server provides a second access log implementation called the *audit log*, which is used to provide detailed information about write operations (add, delete, modify, and modify DN) processed within the server. If the audit log is enabled, the entire content of the change is written to the audit log file (which defaults to `logs/audit`) in LDIF form.

The PingDirectory Server also provides a very rich classification system that can be used to filter the content for access log files. This can be helpful when debugging problems with client applications, because it can restrict log information to operations processed only by a particular application (for example, based on IP address and/or authentication DN), only failed operations, or only operations taking a long time to complete, etc.

Setup log

The **`setup`** tool writes a log file providing information about the processing that it performs. By default, this log file is written to `logs/setup.log` although a different name may be used if a file with that name already exists, because the **`setup`** tool has already been run. The full path to the setup log file is provided when the **`setup`** tool has completed.

Tool log

Many of the administrative tools provided with the Directory Server (for example, **`import-ldif`**, **`export-ldif`**, **`backup`**, **`restore`**, etc.) can take a significant length of time to complete write information to standard output or standard error or both while the tool is running. They also write additional output to files in the `logs/tools` directory (for example, **logs/tools/ import-ldif.log**). The information written to these log files can be useful for diagnosing problems encountered while they were running. When running via the server tasks interface, log messages generated while the task is running may alternately be written to the server error log file.

je.info and je.config files

The primary datastore used by the Directory Server is the Oracle Berkeley DB Java Edition (JE). The Directory Server provides two primary sources of information about processing within the database.

The first is logging performed by the JE code itself, and is written into the `je.info.0` file in the server containing the database files (for example, `db/userRoot/je.info.0`). In the event of a problem within JE itself, useful information about the nature of the problem may be written to this log. The level of information written to this log file is controlled by the `db-logging-level` property in the backend configuration object. It uses the standard Java logging framework for logging messages, so the standard SEVERE, WARNING, INFO, CONFIG, FINE, FINER, and FINEST levels are available.

The second is configuration information used when opening the database environment. When the backend database environment is opened, then the Directory Server will also write a file named `je.config` in the server containing the database files (for example, `db/userRoot/je.config`) with information about the configuration used.

LDAP SDK debug log

This log can be used to help examine the communication between the Directory Server and the Directory Proxy Server. It contains information about exceptions that occur during processing, problems establishing and terminating network connections, and problems that occur during the reading and writing of LDAP messages and LDIF entries. You can configure the types of debugging that should be enabled, the debug level that should be used, and whether debug messages should include stack traces. As for other file-based loggers, you can also specify the rotation and retention policies.

About the monitor entries

While the Directory Server is running, it generates a significant amount of information available through monitor entries. Monitor entries are available over LDAP in the `cn=monitor` subtree. The types of monitor entries that are available include:

- **General Monitor Entry (cn=monitor)** – Provides a basic set of general information about the server.
- **Active Operations Monitor Entry (cn=Active Operations,cn=monitor)** – Provides information about all operations currently in progress in the server.
- **Backend Monitor Entries (cn={id} Backend,cn=monitor)** – Provides information about the backend, including the number of entries, the base DN(s), and whether it is private.
- **Client Connections Monitor Entry (cn=Client Connections,cn=monitor)** – Provides information about all connections currently established to the server.
- **Connection Handler Monitor Entry (cn={name},cn=monitor)** – Provides information about the configuration of each connection handler and the client connections established to it.
- **Database Environment Monitor Entries (cn={id} Database Environment,cn=monitor)** – Provides statistics and other data from the Oracle Berkeley DB Java Edition database envi- ronment used by the associated backend.
- **Disk Space Usage Monitor Entry (cn=Disk Space Usage,cn=monitor)** – Provides infor- mation about the amount of usable disk space available to server components.
- **JVM Memory Usage Monitor Entry (cn=JVM Memory Usage,cn=monitor)** – Provides information about garbage collection activity, the amount of memory available to the server, and the amount of memory consumed by various server components.

- **JVM Stack Trace Monitor Entry (cn=JVM Stack Trace,cn=monitor)** – Provides a stack trace of all threads in the JVM.
- **LDAP Statistics Monitor Entries (cn={name} Statistics,cn=monitor)** – Provides information about the number of each type of operation requested and bytes transferred over the connection handler.
- **Processing Time Histogram Monitor Entry (cn=Processing Time Histogram,cn=monitor)** – Provides information about the number of percent of operations that completed in various response time categories.
- **SSL Context Monitor Entry (cn=SSL Context,cn=monitor)** – Provides information about the available and supported SSL Cipher Suites and Protocols on the server.
- **System Information Monitor Entry (cn=System Information,cn=monitor)** – Provides information about the underlying JVM and system.
- **Version Monitor Entry (cn=Version,cn=monitor)** – Provides information about the Directory Server version.
- **Work Queue Monitor Entry (cn=Work Queue,cn=monitor)** – Provides information about the state of the Directory Server work queue, including the number of operations waiting on worker threads and the number of operations that have been rejected because the queue became full.

Directory Server troubleshooting tools

The PingDirectory Server provides a set of tools that can also be used to obtain information for diagnosing and solving problems.

Server version information

If it becomes necessary to contact your authorized support provider, then it will be important to provide precise information about the version of the Directory Server software that is in use. If the server is running, then this information can be obtained from the "`cn=Version,cn=monitor`" entry. It can also be obtained using the command:

```
$ bin/status --fullVersion
```

This command outputs a number of important pieces of information, including:

- Major, minor, point and patch version numbers for the server.
- Source revision number from which the server was built.
- Build information including build ID with timestamp, OS, user, Java and JVM version for the build.
- Auxiliary software versions: Jetty, JZlib, SNMP4j (SNMP4J, Agent, Agentx), Groovy, LDAP SDK for Java, and the Server SDK.

LDIF connection handler

The Directory Server provides an LDIF connection handler that provides a way to request operations that do not require any network communication with the server. This can be particularly helpful if a configuration problem or bug in the server has left a connection handler unusable, or if all worker threads are busy processing operations.

The LDIF connection handler is enabled by default and looks for LDIF files to be placed in the `config/auto-process-ldif` directory. This Directory Server does not exist by default, but if it is created and an LDIF file is placed in it that contains one or more changes to be processed, then those changes will be applied.

Any changes that can be made over LDAP can be applied through the LDIF connection handler. It is primarily intended for administrative operations like updating the server configuration or scheduling tasks, although other types of changes (including changes to data contained in the server) can be processed. As the LDIF file is processed, a new file is written with comments for each change providing information about the result of processing that change.

dbtest tool

The **dbtest** tool provides a utility that can be used to obtain general information about the data in a backend database. The tool dumps information about entries or keys, and raw data from the database. It can also find keys that have exceeded the entry limit.

For example, the following command can be used to dump a list of all keys in the `objectClass.equality` that have exceeded the entry threshold:

```
$ bin/dbtest dump-database-container \
  --backendID userRoot \
  --baseDN "dc=example,dc=com" \
  --databaseName objectClass.equality \
  --onlyExceedingLimit
```

On a large database, many **dbtest** operations may take a long time to complete, since every record in the associated database is examined. Use the database name option to list a specific database. The following command displays information about the uid.equality database in the `dc=example,dc=com` entry container in the userRoot backend.

```
$ bin/dbtest list-database-containers -n userRoot -b "dc=example,dc=com" -d
 uid.equality
```

Index key entry limit

Indexes have keys that maintain a list of matching entries, up to the index entry limit. When that limit is reached, the key will not contain or maintain that list, and will just maintain a count of matching entries. To determine if index keys are approaching their limit, use either the **dbtest** tool or the **verify-index** tool.

While the **dbtest** tool can be used to gather general imformation, the **verify-index** tool provides statistical data about the percent of entries covered by the keys.

For example, the following command can be used to retrieve a list of keys that have exceeded the entry threshold:

```
$ bin/verify-index \
  --baseDN dc=example,dc=com \
  --listKeysExceedingIndexEntryLimit
```

The following is a sample of the data returned:

```
 [12:06:05]  Checked 6003 entries and found 0 error(s) in 2 seconds (average
rate 2453.2/sec)
 [12:06:05]  Statistics for records that have exceeded the entry limit:
 [12:06:05]  The st.equality index has 48 such record(s) limit=100 min=103
max=152 median=118
 [12:06:05]  1. or (152 entries / 2.53% of all entries)
 [12:06:05]  2. ma (132 entries / 2.20% of all entries)
  .....
 [12:06:05]  The id2subtree index has 2 such record(s) limit=4000 min=6000
max=6002 median=6001
 [12:06:05]  1. 1 => dc=example,dc=com (6002 entries / 99.98% of all entries)
  .....
 [12:06:05]  The id2children index has 1 such record(s) limit=4000 min=6000
max=6000 median=6000
 [12:06:05]  1. 2 => ou=People,dc=example,dc=com (6000 entries / 99.95% of all
entries)
 [12:06:05]  The objectClass.equality index has 4 such record(s) limit=4000
min=6001 max=6003 median=6001
 [12:06:05]  1. top (6003 entries / 100.00% of all entries)
  .....
```

Embedded profiler

If the Directory Server appears to be running slowly, then it is helpful to know what operations are being processed in the server. The JVM Stack Trace monitor entry can be used to obtain a point-in-time snapshot of what the server is doing, but in many cases, it might be useful to have information collected over a period of time.

The embedded profiler is configured so that it is always available but is not active by default so that it has no impact on the performance of the running server. Even when it is running, it has a relatively small impact on performance, but it is recommended that it remain inactive when it is not needed. It can be controlled using the **`dsconfig`** tool or the Administrative Console by managing the "Profiler" configuration object in the "Plugin" object type, available at the standard object level. The `profile-action` property for this configuration object can have one of the following values:

- **start** – Indicates that the embedded profiler should start capturing data in the background.
- **stop** – Indicates that the embedded profiler should stop capturing data and write the information that it has collected to a `logs/profile{timestamp}` file.
- **cancel** – Indicates that the embedded profiler should stop capturing data and discard any information that it has collected.

Any profiling data that has been captured can be examined using the **`profiler-viewer`** tool. This tool can operate in either a text-based mode, in which case it dumps a formatted text representation of the profile data to standard output, or it can be used in a graphical mode that allows the information to be more easily understood.

Invoking the profile viewer in text-based mode

Steps

- Run the **`profile-viewer`** command and specify the captured log file using the `--fileName` option.

```
$ bin/profile-viewer --fileName logs/profile.20110101000000Z
```

Invoking the profile viewer in GUI mode

Steps

- Run the **`profile-viewer`** command and specify the captured log file using the `--fileName` option. To invoke GUI mode, add the option `--useGUI`.

```
$ bin/profile-viewer --fileName logs/profile.20110101000000Z --useGUI
```

Oracle Berkeley DB Java Edition utilities

The Oracle Berkeley DB Java Edition (JE) itself provides a number of utilities that can be used for performing various types of low-level debugging in the database environment. These utilities should generally not be used unless you are advised to do so by your authorized support provider, but they provide access to information about the underlying database environment that is not available through any other means.

Troubleshooting resources for Java applications

Because the PingDirectory Server is written entirely in Java, it is possible to use standard Java debugging and instrumentation tools when troubleshooting problems with the Directory Server. In many cases, obtaining the full benefit of these tools requires access to the Directory Server source code. These Java tools should be used under the advisement of your authorized support provider.

Java troubleshooting tools

The Java Development Kit provides a number of very useful tools to obtain information about Java applications and diagnosing problems. These tools are not included with the Java Runtime Environment

(JRE), so the full Java Development Environment (JDK) should always be installed and used to run the PingDirectory Server.

jps

The **jps** tool is a Java-specific version of the UNIX **ps** tool. It can be used to obtain a list of all Java processes currently running and their respective process identifiers. When invoked by a non-root user, it will list only Java processes running as that user. When invoked by a root user, then it lists all Java processes on the system.

This tool can be used to see if the Directory Server is running and if a process ID has been assigned to it. This process ID can be used in conjunction with other tools to perform further analysis.

This tool can be run without any arguments, but some of the more useful arguments that include:

- **-v** – Includes the arguments passed to the JVM for the processes that are listed.
- **-m** – Includes the arguments passed to the main method for the processes that are listed.
- **-l** – (lowercase L). Include the fully qualified name for the main class rather than only the base class name.

jstack

The **jstack** tool is used to obtain a stack trace of a running Java process, or optionally from a core file generated if the JVM happens to crash. A stack trace can be extremely valuable when trying to debug a problem, because it provides information about all threads running and exactly what each is doing at the point in time that the stack trace was obtained.

Stack traces are helpful when diagnosing problems in which the server appears to be hung or behaving slowly. Java stack traces are generally more helpful than native stack traces, because Java threads can have user-friendly names (as do the threads used by the PingDirectory Server), and the frame of the stack trace may include the line number of the source file to which it corresponds. This is useful when diagnosing problems and often allows them to be identified and resolved quickly.

To obtain a stack trace from a running JVM, use the command:

```
jstack {processID}
```

where {processID} is the process ID of the target JVM as returned by the **jps** command. To obtain a stack trace from a core file from a Java process, use the command:

```
jstack {pathToJava} {pathToCore}
```

where `{pathToJava}` is the path to the java command from which the core file was created, and `{pathToCore}` is the path to the core file to examine. In either case, the stack trace is written to standard output and includes the names and call stacks for each of the threads that were active in the JVM.

In many cases, no additional options are necessary. The "`-l`" option can be added to obtain a long listing, which includes additional information about locks owned by the threads. The "`-m`" option can be used to include native frames in the stack trace.

jmap

The **jmap** tool is used to obtain information about the memory consumed by the JVM. It is very similar to the native **pmap** tool provided by many operating systems. As with the **jstack** tool, **jmap** can be invoked against a running Java process by providing the process ID, or against a core file, like:

```
jmap {processID}
jmap {pathToJava} {pathToCore}
```

Some of the additional arguments include:

- **-dump:live,format=b,file=filename** – Dump the live heap data to a file that can be examined by the **jhat** tool

- **-heap** – Provides a summary of the memory used in the Java heap, along with information about the garbage collection algorithm in use.
- **-histo:live** – Provides a count of the number of objects of each type contained in the heap. If the "`:live`" portion is included, then only live objects are included; otherwise, the count include objects that are no longer in use and are garbage collected.

jhat

The **`jhat`** (Java Heap Analysis Tool) utility provides the ability to analyze the contents of the Java heap. It can be used to analyze a heap dump file, which is generated if the Directory Server encounters an out of memory error (as a result of the "`-XX:+HeapDumpOnOutOfMemoryError`" JVM option) or from the use of the **`jmap`** command with the "`-dump`" option.

The **`jhat`** tool acts as a web server that can be accessed by a browser in order to query the contents of the heap. Several predefined queries are available to help determine the types of objects consuming significant amounts of heap space, and it also provides a custom query language (OQL, the Object Query Language) for performing more advanced types of analysis.

The **`jhat`** tool can be launched with the path to the heap dump file, like:

```
jhat /path/to/heap.dump
```

This command causes the **`jhat`** web server to begin listening on port 7000. It can be accessed in a browser at `http://localhost:7000` (or `http://address:7000` from a remote system). An alternate port number can be specified using the "`-port`" option, like:

```
jhat -port 1234 /path/to/heap.dump
```

To issue custom OQL searches, access the web interface using the URL `http://localhost:7000/oql/` (the trailing slash must be provided). Additional information about the OQL syntax may be obtained in the web interface at `http://localhost:7000/oqlhelp/`.

jstat

The **`jstat`** tool is used to obtain a variety of statistical information from the JVM, much like the **`vmstat`** utility that can be used to obtain CPU utilization information from the operating system. The general manner to invoke it is as follows:

```
jstat {type} {processID} {interval}
```

The `{interval}` option specifies the length of time in milliseconds between lines of output. The `{processID}` option specifies the process ID of the JVM used to run the Directory Server, which can be obtained by running **`jps`** as mentioned previously. The `{type}` option specifies the type of output that should be provided. Some of the most useful types include:

- **-class** – Provides information about class loading and unloading.
- **-compile** – Provides information about the activity of the JIT complex.
- **-printcompilation** – Provides information about JIT method compilation.
- **-gc** – Provides information about the activity of the garbage collector.
- **-gccapacity** – Provides information about memory region capacities.

Java diagnostic information

In addition to the tools listed in the previous section, the JVM can provide additional diagnostic information in response to certain events.

JVM crash diagnostic information

If the JVM itself should happen to crash for some reason, then it generates a fatal error log with information about the state of the JVM at the time of the crash. By default, this file is named `hs_err_pid{processID}.log` and is written into the base directory of the Directory Server installation. This file includes information on the underlying cause of the JVM crash, information about the threads

running and Java heap at the time of the crash, the options provided to the JVM, environment variables that were set, and information about the underlying system.

Troubleshooting resources in the operating system

The underlying operating system also provides a significant amount of information that can help diagnose issues that impact the performance and the stability of the Directory Server. In some cases, problems with the underlying system can be directly responsible for the issues seen with the Directory Server, and in others system, tools can help narrow down the cause of the problem.

Identifying problems with the underlying system

If the underlying system itself is experiencing problems, it can adversely impact the function of applications running on it. To look for problems in the underlying system view the system log file (`/var/log/messages` on Linux). Information about faulted or degraded devices or other unusual system conditions are written there.

Examining CPU utilization

Observing CPU utilization for the Directory Server process and the system as a whole provides clues as to the nature of the problem.

System-Wide CPU utilization

To investigate CPU consumption of the system as a whole, use the **vmstat** command with a time interval in seconds, like:

```
vmstat 5
```

The specific output of this command varies between different operating systems, but it includes the percentage of the time the CPU was spent executing user-space code (user time), the percentage of time spent executing kernel-space code (system time), and the percentage of time not executing any code (idle time).

If the CPUs are spending most of their time executing user-space code, the available processors are being well-utilized. If performance is poor or the server is unresponsive, it can indicate that the Directory Server is not optimally tuned. If there is a high system time, it can indicate that the system is performing excessive disk and/or network I/O, or in some cases, there can be some other system-wide problem like an interrupt storm. If the system is mostly idle but the Directory Server is performing poorly or is unresponsive, there can be a resource constraint elsewhere (for example, waiting on disk or memory access, or excessive lock contention), or the JVM can be performing other tasks like stop-the-world garbage collection that cannot be run heavily in parallel.

Per-CPU utilization

To investigate CPU consumption on a per-CPU basis, use the **mpstat** command with a time interval in seconds, like:

```
mpstat 5
```

On Linux systems, it might be necessary to add "`-P ALL`" to the command, like:

```
mpstat -P ALL 5
```

Among other things, this shows the percentage of time each CPU has spent in user time, system time, and idle time. If the overall CPU utilization is relatively low but **mpstat** reports that one CPU has a much higher utilization than the others, there might be a significant bottleneck within the server or the JVM might be performing certain types of garbage collection which cannot be run in parallel. On the other hand, if CPU utilization is relatively even across all CPUs, there is likely no such bottleneck and the issue might be elsewhere.

Per-process utilization

To investigate CPU consumption on a per-process basis, use a command such as the **top**Directory Server is consuming a significant amount of available CPU, it might be interfering with the ability of the Directory Server to run effectively.

Examining disk utilization

If the underlying system has a very high disk utilization, it can adversely impact Directory Server performance. It could delay the ability to read or write database files or write log files. It could also raise concerns for server stability if excessive disk I/O inhibits the ability of the cleaner threads to keep the database size under control.

The **iostat** tool may be used to obtain information about the disk activity on the system.

On Linux systems, **iostat** should be invoked with the "`-x`" argument, like:

```
iostat -x 5
```

A number of different types of information will be displayed, but to obtain an initial feel for how busy the underlying disks are, look at the "%util" column on Linux. This field shows the percentage of the time that the underlying disks are actively servicing I/O requests. A system with a high disk utilization likely exhibits poor Directory Server performance.

If the high disk utilization is on one or more disks that are used to provide swap space for the system, the system might not have enough free memory to process requests. As a result, it might have started swapping blocks of memory that have not been used recently to disk. This can cause very poor server performance. It is important to ensure that the server is configured appropriately to avoid this condition. If this problem occurs on a regular basis, then the server is likely configured to use too much memory. If swapping is not normally a problem but it does arise, then check to see if there are any other processes running, which are consuming a significant amount of memory, and check for other potential causes of significant memory consumption (for example, large files in a **tmpfs** file system).

Examining process details

There are a number of tools provided by the operating system that can help examine a process in detail.

ps

The standard **ps** tool can be used to provide a range of information about a particular process. For example, the command can be used to display the state of the process, the name of the user running the process, its process ID and parent process ID, the priority and nice value, resident and virtual memory sizes, the start time, the execution time, and the process name with arguments:

```
ps -fly -p {processID}
```

Note that for a process with a large number of arguments, the standard **ps** command displays only a limited set of the arguments based on available space in the terminal window.

pstack

The **pstack** command can be used to obtain a native stack trace of all threads in a process. While a native stack trace might not be as user-friendly as a Java stack trace obtained using **jstack**, it includes threads that are not available in a Java stack trace. For example, the command displays those threads used to perform garbage collection and other housekeeping tasks. The general usage for the **pstack** command is:

```
pstack {processID}
```

dbx / gdb

A process debugger provides the ability to examine a process in detail. Like **pstack**, a debugger can obtain a stack trace for all threads in the process, but it also provides the ability to examine a process (or core file) in much greater detail, including observing the contents of memory at a specified address and the

values of CPU registers in different frames of execution. The GNU debugger **gdb** is widely-used on Linux systems.

Note that using a debugger against a live process interrupts that process and suspends its execution until it detaches from the process. In addition, when running against a live process, a debugger has the ability to actually alter the contents of the memory associated with that process, which can have adverse effects. As a result, it is recommended that the use of a process debugger be restricted to core files and only used to examine live processes under the direction of your authorized support provider.

pfiles / lsof

To examine the set of files that a process is using (including special types of files, like sockets), you can use a tool such as **lsof** on Linux systems, (

```
lsof -p {processID}
```

)

Tracing process execution

If a process is unresponsive but is consuming a nontrivial amount of CPU time, or if a process is consuming significantly more CPU time than is expected, it might be useful to examine the activity of that process in more detail than can be obtained using a point-in-time snapshot. For example, if a process is performing a significant amount of disk reads and/or writes, it can be useful to see which files are being accessed. Similarly, if a process is consistently exiting abnormally, then beginning tracing for that process just before it exits can help provide additional information that cannot be captured in a core file (and if the process is exiting rather than being terminated for an illegal operation, then no core file may be available).

This can be accomplished using the **strace** tool on Linux (

```
strace -f -p {processID}
```

).

Consult the **strace** manual page for additional information.

Problems with SSL communication

Enable TLS debugging in the server to troubleshoot SSL communication issues:

```
$ dsconfig create-debug-target \
  --publisher-name "File-Based Debug Logger" \
  --target-name
 com.unboundid.directory.server.extensions.TLSConnectionSecurityProvider \
  --set debug-level:verbose \
  --set include-throwable-cause:true
```

```
$ dsconfig set-log-publisher-prop \
  --publisher-name "File-Based Debug Logger" \
  --set enabled:true \
  --set default-debug-level:disabled
```

In the `java.properties` file, add `-Djavax.net.debug=ssl` to the `start-server` line, and run `bin/dsjavaproperties` to make the option take effect on a scheduled server restart.

Examining network communication

Because the PingDirectory Server is a network-based application, it can be valuable to observe the network communication that it has with clients. The Directory Server itself can provide details about its interaction with clients by enabling debugging for the protocol or data debug categories, but there may be a number of cases in which it is useful to view information at a much lower level. A network sniffer, like the *tcpdump* tool on Linux, can be used to accomplish this.

There are many options that can be used with these tools, and their corresponding manual pages will provide a more thorough explanation of their use. However, to perform basic tracing to show the full details of the packets received for communication on port 389 with remote host 1.2.3.4, the following command can be used on Linux:

```
tcpdump -i {interface} -n -XX -s 0 host 1.2.3.4 and port 389
```

It does not appear that the **tcpdump** tool provides support for LDAP parsing. However, it is possible to write capture data to a file rather than displaying information on the terminal (using "`-w {path}`" with **tcpdump**), so that information can be later analyzed with a graphical tool like Wireshark, which provides the ability to interpret LDAP communication on any port.

Note that enabling network tracing generally requires privileges that are not available to normal users and therefore may require root access.

Common problems and potential solutions

This section describes a number of different types of problems that can occur and common potential causes for them.

General troubleshooting methodology

When a problem is detected, Ping Identity recommends using the following general methodology to isolate the problem:

1. Run the **bin/status** tool or look at the server status in the Administrative Console. The **status** tool provides a summary of the server's current state with key metrics and a list of recent alerts.
2. Look in the server logs. In particular, view the following logs:
   - logs/errors
   - logs/failed-ops
   - logs/expensive-ops
3. Use system commands, such as **vmstat** and **iostat** to determine if the server is bottle-necked on a system resource like CPU or disk throughput.
4. For performance problem (especially intermittent ones like spikes in response time), enabling the **periodic-stats-logger** can help to isolate problems, because it stores important server performance information on a per-second basis. The **periodic-stats-logger** can save the information in a csv-formatted file that can be loaded into a spreadsheet. The information this logger makes available is very configurable. You can create multiple loggers for different types of information or a different frequency of logging (for example, hourly data in addition to per-second data). For more information, see "Profiling Server Performance Using the Periodic Stats Logger".
5. For replication problem, run **dsreplication status** and look at the `logs/replication` file.
6. For more advanced users, run the **collect-support-data** tool on the system, unzip the archive somewhere, and look through the collected information. This is often useful when administrators most familiar with the PingData Platform do not have direct access to the systems where the production servers are running. They can examine the **collect-support-data** archive on a different server. For more information, see Using the Collect Support Data Tool.

ⓘ **Important:** Run the **collect-support-data** tool whenever there is a problem whose cause is not easily identified, so that this information can be passed back to your authorized support provider before corrective action can be taken.

The Server will not run setup

If the **setup** tool does not run properly, some of the most common reasons include the following:

A suitable Java environment is not available

The PingDirectory Server requires that Java be installed on the system and made available to the server, and it must be installed prior to running **setup**. If the **setup** tool does not detect that a suitable Java environment is available, it will refuse to run.

To ensure that this does not happen, the **setup** tool should be invoked with an explicitly-defined value for the *JAVA_HOME* environment variable that specifies the path to the Java installation that should be used. For example:

```
env JAVA_HOME=/ds/java ./setup
```

If this still does not work for some reason, then it can be that the value specified in the provided *JAVA_HOME* environment variable can be overridden by another environment variable. If that occurs, try the following command, which should override any other environment variables that can be set:

```
env UNBOUNDID_JAVA_HOME="/ds/java" UNBOUNDID_JAVA_BIN="" ./setup
```

Oracle Berkeley DB Java Edition is not available

If the version of the Directory Server that you are using was not provided with the Oracle Berkeley DB Java Edition library, then it must be manually downloaded and the appropriate JAR file placed in the `lib` directory before running setup. See the `lib/downloading-je.txt` file for instructions on obtaining the appropriate library.

Unexpected arguments provided to the JVM

If the **setup** script attempts to launch the java command with an invalid set of Java arguments, it might prevent the JVM from starting. By default, no special options are provided to the JVM when running setup, but this might not be the case if either the *JAVA_ARGS* or *UNBOUNDID_JAVA_ARGS* environment variable is set. If the **setup** tool displays an error message that indicates that the Java environment could not be started with the provided set of arguments, then invoke the following command before trying to re-run **setup**:

```
unset JAVA_ARGS UNBOUNDID_JAVA_ARGS
```

The Server has already been configured or used

The **setup** tool is only intended to provide the initial configuration for the Directory Server. It refuses to run if it detects that the **setup** tool has already been run, or if an attempt has been made to start the Directory Server prior to running the **setup** tool. This protects an existing Directory Server installation from being inadvertently updated in a manner that could harm an existing configuration or data set.

If the Directory Server has been previously used and if you want to perform a fresh installation, it is recommended that you first remove the existing installation, create a new one and run **setup** in that new installation. However, if you are confident that there is nothing of value in the existing installation (for example, if a previous attempt to run **setup** failed to complete successfully for some reason but it will refuse to run again), the following steps can be used to allow the **setup** program to run:

- Remove the `config/config.ldif` file and replace it with the `config/update/config.ldif.{revision}` file containing the initial configuration.
- If there are any files or subdirectories below the `db` directory, then remove them.
- If a `config/java.properties` file exists, then remove it.
- If a `lib/setup-java-home` script (or file on Microsoft Windows) exists, then remove it.

The Server will not start

If the Directory Server does not start, then there are a number of potential causes.

The Server or other administrative tool is already running

Only a single instance of the Directory Server can run at any time from the same installation root. If an instance is already running, then subsequent attempts to start the server will fail. Similarly, some other

administrative operations can also prevent the server from being started. In such cases, the attempt to start the server should fail with a message like:

```
The Directory Server could not acquire an exclusive lock on file
/ds/PingDirectory/locks/server.lock: The exclusive lock requested for file
/ds/PingDirectory/locks/ server.lock was not granted, which indicates
that another process already holds a shared or exclusive lock on that
file. This generally means that another instance of this server is already
running
```

If the Directory Server is not running (and is not in the process of starting up or shutting down) and there are no other tools running that could prevent the server from being started, and the server still believes that it is running, then it is possible that a previously-held lock was not properly released. In that case, you can try removing all of the files in the `locks` directory before attempting to start the server.

If you wish to have multiple instances running at the same time on the same system, then you should create a completely separate installation in another location on the file system.

There is not enough memory available

When the Directory Server is started, the JVM attempts to allocate all memory that it has been configured to use. If there is not enough free memory available on the system, then the Directory Server generates an error message that indicates that the server could not be started with the specified set of arguments. Note that it is possible that an invalid option was provided to the JVM (as described below), but if that same set of JVM arguments has already been used successfully to run the server, then it is more likely that the system does not have enough memory available.

There are a number of potential causes for this:

- If the amount of memory in the underlying system has changed (for example, system memory has been removed, or if the Directory Server is running in a zone or other type of virtualized container and a change has been made to the amount of memory that container will be allowed to use), then the Directory Server might need to be re-configured to use a smaller amount of memory than had been previously configured.
- Another process running on the system is consuming a significant amount of memory so that there is not enough free memory available to start the server. If this is the case, then either terminate the other process to make more memory available for the Directory Server, or reconfigure the Directory Server to reduce the amount of memory that it attempts to use.
- The Directory Server was just shut down and an attempt was made to immediately restart it. In some cases, if the server is configured to use a significant amount of memory, then it can take a few seconds for all of the memory that had been in use by the server, when it was previously running, to be released back to the operating system. In that case, run the **`vmstat`** command and wait until the amount of free memory stops growing before attempting to restart the server.
- If the system is configured with one or more memory-backed file systems, verify whether any large files might be consuming a significant amount of memory in any of those locations. If so, remove them or relocate them to a disk-based file system.
- For Linux systems only, if a mismatch exists between the huge pages setting for the JVM and the huge pages reserved in the operating system.

If nothing else works and there is still not enough free memory to allow the JVM to start, then as a last resort, try rebooting the system.

An invalid Java Environment or JVM option was used

If an attempt to start the Directory Server fails with an error message indicating that no valid Java environment could be found, or indicates that the Java environment could not be started with the configured set of options, then you should first ensure that enough memory is available on the system as described above. If there is a sufficient amount of memory available, then other causes for this error can include the following:

- The Java installation that was previously used to run the server no longer exists (for example, an updated Java environment was installed and the old installation was removed). In that case, update the `config/java.properties` file to reference to path to the new Java installation and run the `bin/dsjavaproperties` command to apply that change.
- The Java installation used to run the server has been updated and the server is trying to use the correct Java installation but one or more of the options that had worked with the previous Java version no longer work with the new version. In that case, it is recommended that the server be re-configured to use the previous Java version, so that it can be run while investigating which options should be used with the new installation.
- If an *UNBOUNDID_JAVA_HOME* or *UNBOUNDID_JAVA_BIN* environment variable is set, then its value may override the path to the Java installation used to run the server as defined in the `config/java.properties` file. Similarly, if an *UNBOUNDID_JAVA_ARGS* environment variable is set, then its value might override the arguments provided to the JVM. If this is the case, then explicitly unset the *UNBOUNDID_JAVA_HOME*, *UNBOUNDID_JAVA_BIN*, and *UNBOUNDID_JAVA_ARGS* environment variables before trying to start the server.

Note that any time the `config/java.properties` file is updated, the **`bin/dsjavaproperties`** tool must be run to apply the new configuration. If a problem with the previous Java configuration prevents the **`bin/dsjavaproperties`** tool from running properly, then it can be necessary to remove the `lib/set-java-home` script (or `lib\set-java-home.bat` file on Microsoft Windows) and invoke the **`bin/dsjavaproperties`** tool with an explicitly-defined path to the Java environment, like:

```
env UNBOUNDID_JAVA_HOME=/ds/java bin/dsjavaproperties
```

An invalid command-line option was provided

There are a small number of arguments that are provided when running the **`bin/start-server`** command, but in most cases, none are required. If one or more command-line arguments were provided for the **`bin/start-server`** command and any of them is not recognized, then the server provides an error message indicating that an argument was not recognized and displays version information. In that case, correct or remove the invalid argument and try to start the server again.

The Server has an invalid configuration

If a change is made to the Directory Server configuration using an officially-supported tool like **`dsconfig`** or the Administrative Console, the server should validate that configuration change before applying it. However, it is possible that a configuration change can appear to be valid at the time that it is applied, but does not work as expected when the server is restarted. Alternately, a change in the underlying system can cause a previously-valid configuration to become invalid.

In most cases involving an invalid configuration, the Directory Server displays (and writes to the error log) a message that explains the problem, and this can be sufficient to identify the problem and understand what action needs to be taken to correct it. If for some reason the startup failure does not provide enough information to identify the problem with the configuration, then look in the `logs/config-audit.log` file to see what recent configuration changes have been made with the server online, or in the `config/archived-configs` directory to see if there might have been a recent configuration change resulting from a direct change to the configuration file itself that was not made through a supported configuration interface.

If the server does not start as a result of a recent invalid configuration change, then it can be possible to start the server using the configuration that was in place the last time that the server started successfully (for example, the "last known good" configuration). This can be achieved using the `--useLastKnownGoodConfig` option:

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

Note that if it has been a long time since the last time the server was started and a number of configuration changes have been made since that time, then the last known good configuration can be significantly out of date. In such cases, it can be preferable to manually repair the configuration.

If there is no last known good configuration, if the server no longer starts with the last known good configuration, or if the last known good configuration is significantly out of date, then manually update the configuration by editing the `config/config.ldif` file. In that case, you should make sure that the server is offline and that you have made a copy of the existing configuration before beginning. You might wish to discuss the change with your authorized support representative before applying it to ensure that you understand the correct change that needs to be made.

ⓘ **Note:** In addition to manually-editing the config file, you can look at previous achived configurations to see if the most recent one works. You can also use the **`ldif-diff`** tool to compare the configurations in the archive to the current configuration to see what is different.

You do not have sufficient permissions

The Directory Server should only be started by the user or role used to initially install the server. In most cases, if an attempt is made to start the server as a user or role other than the one used to create the initial configuration, then the server will fail to start, because the user will not have sufficient permissions to access files owned by the other user, such as database and log files. However, if the server was initially installed as a non-root user and then the server is started by the root account, then it can no longer be possible to start the server as a non-root user because new files that are created would be owned by root and could not be written by other users.

If the server was inadvertently started by root when it is intended to be run by a non-root user, or if you wish to change the user account that should be used to run the server, then it should be sufficient to simply change ownership on all files in the Directory Server installation, so that they are owned by the user or role under which the server should run. For example, if the Directory Server should be run as the "ds" user in the "other" group, then the following command can be used to accomplish this (invoked by the root user):

```
chown -R ds:other /ds/PingDirectory
```

The Server has crashed or shut itself down

You can first check the current server state by using the **`bin/server-state`** command. If the Directory Server was previously running but is no longer active, then the potential reasons include the following:

- The Directory Server was shut down by an administrator. Unless the server was forcefully terminated (for example, using “kill -9”), then messages are written to the `error` and `server.out` logs explaining the reason for the shutdown.
- The Directory Server was shut down when the underlying system crashed or was rebooted. If this is the case, then running the **`uptime`** command on the underlying system shows that it was recently booted.
- The Directory Server process was terminated by the underlying operating system for some reason (for example, the out of memory killer on Linux). If this happens, then a message will be written to the system error log.
- The Directory Server decided to shut itself down in response to a serious problem that had arisen. At present, this should only occur if the server has detected that the amount of usable disk space has become critically low, or if significant errors have been encountered during processing that left the server without any remaining worker threads to process operations. If this happens, then messages are written to the `error` and `server.out` logs (if disk space is available) to provide the reason for the shutdown.
- The JVM in which the Directory Server was running crashed. If this happens, then the JVM should dump a fatal error log (a `hs_err_pid{processID}.log` file) and potentially a core file.

In the event that the operating system itself crashed or terminated the process, then you should work with your operating system vendor to diagnose the underlying problem. If the JVM crashed or the server shut itself down for a reason that is not clear, then contact your authorized support provider for further assistance.

Conditions for automatic server shutdown

All PingDirectory Server servers will shutdown in an out of memory condition, a low disk space error state, or for running out of file descriptors. The Directory Server will enter lockdown mode on unrecoverable database environment errors, but can be configured to shutdown instead with this setting:

```
$ dsconfig set-global-configuration-prop \
            --set unrecoverable-database-error-mode:initiate-server-shutdown
```

The Server will not accept client connections

You can first check the current server state by using the **bin/server-state** command. If the Directory Server does not appear to be accepting connections from clients, then potential reasons include the following:

- The Directory Server is not running.
- The underlying system on which the Directory Server is installed is not running.
- The Directory Server is running but is not reachable as a result of a network or firewall configuration problem. If that is the case, then connection attempts should time out rather than be rejected.
- If the Directory Server is configured to allow secure communication via SSL or StartTLS, then a problem with the key manager and/or trust manager configuration can cause connections to be rejected. If that is the case, then messages should be written to the server access log for each failed connection attempt.
- If the Directory Server has been configured with a maximum allowed number of connections, then it can be that the maximum number of allowed client connections are already established. If that is the case, then messages should be written to the server access log for each rejected connection attempt.
- If the Directory Server is configured to restrict access based on the address of the client, then messages should be written to the server access log for each rejected connection attempt.
- If a connection handler encounters a significant error, then it can stop listening for new requests. If this occurs, then a message should be written to the server error log with information about the problem. Another solution is to restart the server. A third option is to restart the connection handler using the LDIF connection handler to make it available again. To do this, create an LDIF file that disables and then re-enables the connection handler, create the `config/auto-process-ldif` directory if it does not already exist, and then copy the LDIF file into it.

The Server is unresponsive

You can first check the current server state by using the **bin/server-state** command. If the Directory Server process is running and appears to be accepting connections but does not respond to requests received on those connections, then potential reasons for this behavior include:

- If all worker threads are busy processing other client requests, then new requests that arrive will be forced to wait in the work queue until a worker thread becomes available. If this is the case, then a stack trace obtained using the **jstack** command shows that all of the worker threads are busy and none of them are waiting for new requests to process.

  A dedicated thread pool can be used for processing administrative operations. This thread pool enables diagnosis and corrective action if all other worker threads are processing operations. To request that operations use the administrative thread pool, using the **ldapsearch** command for example, use the **--useAdministrativeSession** option. The requester must have the **use-admin-session** privilege (included for root users). By default, eight threads are available for this purpose. This can be changed with the **num-administrative-session-worker-threads** property in the work queue configuration.

> ⓘ **Note:** If all of the worker threads are tied up processing the same operation for a long time, the server will also issue an alert that it might be deadlocked, which may not actually be the case. All threads might be tied up processing unindexed searches.

- If a request handler is stuck performing some expensive processing for a client connection, then other requests sent to the server on connections associated with that request handler is forced to wait until the request handler is able to read data on those connections. If this is the case, then only some of the connections can experience this behavior (unless there is only a single request handler, in which it will impact all connections), and stack traces obtained using the **jstack** command shows that a request handler thread is continuously blocked rather than waiting for new requests to arrive. Note that this scenario is a theoretical problem and one that has not appeared in production.
- If the JVM in which the Directory Server is running is not properly configured, then it can be forced to spend a significant length of time performing garbage collection, and in severe cases, could cause significant interruptions in the execution of Java code. In such cases, a stack trace obtained from a **pstack** of the native process should show that most threads are idle but at least one thread performing garbage collection is active. It is also likely that one or a small number of CPUs is 100% busy while all other CPUs are mostly idle. The server will also issue an alert after detecting a long JVM pause (due to garbage collection). The alert will include details of the pause.
- If the JVM in which the Directory Server is running has hung for some reason, then the **pstack** utility should show that one or more threads are blocked and unable to make progress. In such cases, the system CPUs should be mostly idle.
- If a network or firewall configuration problem arises, then attempts to communicate with the server cannot be received by the server. In that case, a network sniffer like **snoop** or **tcpdump** should show that packets sent to the system on which the Directory Server is running are not receiving TCP acknowledgement.
- If the system on which the Directory Server is running has become hung or lost power with a graceful shutdown, then the behavior is often similar to that of a network or firewall configuration problem.

If it appears that the problem is with the Directory Server software or the JVM in which it is running, then you need to work with your authorized support provider to fully diagnose the problem and determine the best course of action to correct it.

The Server is slow to respond to client requests

If the Directory Server is running and does respond to clients, but clients take a long time to receive responses, then the problem can be attributable to a number of potential problems. In these cases, use the Periodic Stats Logger, which is a valuable tool to get per-second monitoring information on the Directory Server. The Periodic Stats Logger can save the information in csv format for easy viewing in a spreadsheet. For more information, see "Profiling Server Performance Using the Periodic Stats Logger". The potential problems that cause slow responses to client requests are as follows:

- The server is not optimally configured for the type of requests being processed, or clients are requesting inefficient operations. If this is the case, then the access log should show that operations are taking a long time to complete and they will likely be unindexed. In that case, updating the server configuration to better suit the requests, or altering the requests to make them more efficient, could help alleviate the problem. In this case, view the expensive operations access log in `logs/expensive-ops`, which by default logs operations that take longer than 1 second. You can also run the **bin/status** command or view the status in the Administrative Console to see the Directory Server's Work Queue information (also see the next bullet point).
- The server is overwhelmed with client requests and has amassed a large backlog of requests in the work queue. This can be the result of a configuration problem (for example, too few worker thread configured), or it can be necessary to provision more systems on which to run the Directory Server software. Symptoms of this problem appear similar to those experienced when the server is asked to process inefficient requests, but looking at the details of the requests in the access log show that they are not necessarily inefficient requests. Run the **bin/status** command to view the Work Queue information. If everything is performing well, you should not see a large queue size or a server that is near 100% busy. The %Busy statistic is calculated as the percentage of worker threads that are busy processing operations.

```
           --- Work Queue ---
           : Recent : Average : Maximum
-----------:--------:---------:--------
```

```
Queue Size : 10  : 1       : 10
% Busy     : 17     : 14      : 100
```

You can also view the expensive operations access log in `logs/expensive-ops`, which by default logs operations that take longer than 1 second.

- The server is not configured to fully cache all of the data in the server, or the cache is not yet primed. In this case, **`iostat`** reports a very high disk utilization. This can be resolved by configuring the server to fully cache all data, and to load database contents into memory on startup. If the underlying system does not have enough memory to fully cache the entire data set, then it might not be possible to achieve optimal performance for operations that need data which is not contained in the cache. For more information, see Disk-Bound Deployments.
- If the JVM is not properly configured, then it will need to perform frequent garbage collection and periodically pause execution of the Java code that it is running. In that case, the server error log should report that the server has detected a number of pauses and can include tuning recommendations to help alleviate the problem.
- If the Directory Server is configured to use a large percentage of the memory in the system, then it is possible that the system has gotten low on available memory and has begun swapping. In this case, **`iostat`** should report very high utilization for disks used to hold swap space, and commands like **`cat /proc/meminfo`** on Linux can report a large amount of swap memory in use. Another cause of swapping is if swappiness is not set to 0 on Linux. For more information, see Disable File System Swapping.
- If another process on the system is consuming a significant amount of CPU time, then it can adversely impact the ability of the Directory Server to process requests efficiently. Isolating the processes (for example, using processor sets) or separating them onto different systems can help eliminate this problem.

The Server returns error responses to client requests

If a large number of client requests are receiving error responses, then view the `logs/failed-ops` log, which is an access log for only failed operations. The potential reasons for the error responses include the following:

- If clients are requesting operations that legitimately should fail (for example, they are targeting entries that do not exist, are attempting to update entries in a way that would violate the server schema, or are performing some other type of inappropriate operation), then the problem is likely with the client and not the server.
- If a portion of the Directory Server data is unavailable (for example, because an online LDIF import or restore is in progress), then operations targeting that data will fail. Those problems will be resolved when the backend containing that data is brought back online. During the outage, it might be desirable to update proxies or load balancers or both to route requests away from the affected server. As of Directory Server version 3.1 or later, the Directory Server will indicate that it is in a degraded status and the Directory Proxy Server will route around it.
- If the Directory Server work queue is configured with a maximum capacity and that capacity has been reached, then the server begins rejecting all new requests until space is available in the work queue. In this case, it might be necessary to alter the server configuration or the client requests or both, so that they can be processed more efficiently, or it might be necessary to add additional server instances to handle some of the workload.
- If an internal error occurs within the server while processing a client request, then the server terminates the connection to the client and logs a message about the problem that occurred. This should not happen under normal circumstances, so you will need to work with your authorized support provider to diagnose and correct the problem.
- If a problem is encountered while interacting with the underlying database (for example, an attempt to read from or write to disk failed because of a disk problem or lack of available disk space), then it can begin returning errors for all attempts to interact with the database until the backend is closed and re-opened and the database has been given a change to recover itself. In these cases, the `je.info.*` file in the database directory should provide information about the nature of the problem.

The Server must disconnect a client connection

If a client connection must be disconnected due to the expense of the client's request, such as an unindexed search across a very large database, perform the following:

- Find the client's connection ID by looking in the **cn=Active Operations,cn=monitor monitor** entry.

```
$ bin/ldapsearch -baseDN cn=monitor "cn=active operations" \
  --bindDN "cn=directory manager"  \
  --bindPassword password
```

- The monitor entry will contain attribute values for **operation-in-progress**, which look like an access log message. Look for the value of **conn** in the client request that should be disconnected. In the following example, the client to be disconnected is requesting a search for **(description=expensive)**, which is on connection 6.

```
dn: cn=Active Operations,cn=monitor
objectClass: top
objectClass: ds-monitor-entry
objectClass: ds-active-operations-monitor-entry
objectClass: extensibleObject
cn: Active Operations
num-operations-in-progress: 2
operation-in-progress: [15/Dec/2014:10:55:35 -0600] SEARCH conn=6 op=3
 msgID=4
     clientIP="10.8.4.21" authDN="cn=app1,ou=applications,dc=example,dc=com"
 base="dc
     =example,dc=com" scope=wholeSubtree filter="(description=expensive)"
 attrs="A
     LL" unindexed=true
operation-in-progress: [15/Dec/2014:10:56:11 -0600] SEARCH conn=7 op=1
 msgID=2
     clientIP="127.0.0.1" authDN="cn=Directory Manager,cn=Root
 DNs,cn=config" base="c
     n=monitor" scope=wholeSubtree filter="(cn=active operations)"
 attrs="ALL"
     num-persistent-searches-in-progress: 0
```

- With the connection ID value, create a file with the following contents, named **disconnect6.ldif**.

```
dn: ds-task-id=disconnect6,cn=scheduled Tasks,cn=tasks
objectClass: top
objectClass: ds-task
objectClass: ds-task-disconnect
ds-task-disconnect-connection-id: 6
ds-task-id: disconnect6
ds-task-class-name:
 com.unboundid.directory.server.tasks.DisconnectClientTask
```

- This LDIF file represents a task entry. The connection ID value 6 is assigned to **ds-task-disconnect-connection-id**. The value for **ds-task-id** value does not follow a specific convention. It must be unique among other task entries currently cached by the server.
- Disconnect the client and cancel the associated operation by adding the task entry to the server:

```
$ bin/ldapmodify --filename disconnect6.ldif  \
  --defaultAdd --bindDN "cn=directory manager" \
  --bindPassword password
```

The Server is experiencing problems with replication

If replication does not appear to be functioning properly, then first check the **dsreplication status** command, which shows all of the servers that are replicating and whether they are back-logged or not. Next, you can check the server error log, replication repair log, and replication monitor entries may provide

information about the nature of the underlying problem. Potential reasons that replication may not be functioning as expected include the following:

- Replication has not yet been configured between systems or has been disabled.
- If a server has been offline for a period of time or has fallen far enough behind such that it is missing changes, which are no longer present in any of the replication databases, then that server must be re-initialized with an up-to-date copy of the data from another server.
- If the environment is comprised of a heterogeneous set of systems, then it is possible that some of the systems might not be able to keep up with the maximum throughput achieved by other servers in the topology. In such cases, the slower servers might not be fast enough to remain in sync with the other servers in the environment.
- If the environment contains systems in multiple data centers and the network links between the data centers are insufficient for the volume of changes that must be processed, then servers might not be able to remain in sync under a high volume of changes.
- A network or firewall configuration problem has arisen, which prevents or interferes with communication between servers.
- An internal problem within the server has caused replication to stop functioning properly. The Directory Server logs the event in the error log in this case. Run the **`collect-support-data`** tool, so that the details of the problems can be passed to your authorized support provider. Then, try restarting the Directory Server.

How to regenerate the Server ads-certificate

At setup time, the server generates a private key and certificate for use when secure communication between servers is required. This certificate, `ads-certificate`, is stored in `config/ads-truststore` and should typically remain unchanged for the life of the server deployment. If the need arises for a new ads-certificate to be created, say because the server-root has been copied to a new host, then the private key and certificate will be recreated by the startup process if the `config/ads-truststore` and `config/ads-truststore.pin` files are first manually removed while the server is offline. Note that if replication is enabled, the server must have replication disabled before regeneration of the ads-certificate.

For example, the Directory Server allows easy copying of its installation, which can then be used to install another server instance. If a server (ldap1.example.com:389) is enabled with its own copy (ldap2.example.com:389), **`dsreplication`** will exit with the following error message:

```
Replication cannot be enabled between servers ldap1.example.com:389 and
 ldap2.example.com:389
because they are using the same instance key.
```

The solution is to stop the server, remove `config/adstruststore` and `config/adstruststore.pin` and re-start the server. Upon startup, a new `adstruststore`, containing the server's instance key, will be generated. Then, you can re-run **`dsreplication enable`** to set up replication between the two servers.

The Server behaves differently from Sun/Oracle

After migrating from a Sun/Oracle configuration to a PingDirectory Server, follow the tuning procedures in *Sun/Oracle Compatibility* if the Directory Server behaves differently from the Sun/Oracle server.

Troubleshooting ACI evaluation

The Directory Server provides the ability to collect debug information related to ACI evaluation for any operation by enabling the Debug ACI Logger. The Debug ACI Logger is highly configurable and can be scoped to trace very specific request operations in order to narrow on any ACI issue that may arise in the field. Parameters for non-request operations, such as `log-connects`, `log-disconnects`, `log-security-negotiation`, `log-results`, `log-assuance-completed`, `log-search-entries`, `log-search-references`, `log-intermediate-responses` are set to `false` by default and should remain so.

Here is an example to enable the Debug ACI Logger:

```
$ bin/dsconfig set-log-publisher-prop \
  --publisher-name "Debug ACI Logger" \
```

```
  --set enabled:true
```

Using this debug tracer is often more efficient by limiting the output using request and result criteria to match specific types of operations. An example result criteria for operations that fail due to insufficient access rights can be added to the logger as follows:

```
$ bin/dsconfig set-log-publisher-prop \
  --publisher-name "Debug ACI Logger" \
  --set "result-criteria:Insufficient Access Rights"
```

Once the logger is enabled, all matching operations begin writing ACI evaluation traces to the log file. The amount of information is quite large for each evaluation that is done. However, this information is useful if there is an ACI issue that is difficult to resolve. Most operations result in multiple "ACI DEBUG" traces in the log, since it usually requires multiple ACI rights to perform an operation, each of which requires a separate evaluation. In particular, you can expect a lot of debug tracing when dealing with ACIs for controls, extended operations, and proxied authorization.

The ACI DEBUG traces contain the following pieces of information:

- **Operation**. Specifies a dump of the operation object that you can use to correlate to the original request operation.
- **ACI Container**. Specifies the context of the ACI evaluation being performed.
  - **Client Entry**. Specifies an LDIF dump of the client request access.
  - **Resource Entry**. Specifies an LDIF dump of the target resource.
  - **isProxiedAuth**. Specifies if the client is attempting to proxy as another user.
  - **Original Auth**. Specifies the original client DN if authorization is currently via the proxy.
  - **Rights**. Specifies a list of the ACI rights being requested on the resource entry.
  - **Control**. Specifies the OIDs when evaluating ACIs for a control.
  - **ExtOp**. Specifies the OIDs when evaluating ACIs for an extended operation.
- **ACI Canidates**. Specifies a list of all the ACIs known to this operation, sorted by origin.
- **Applicable ACIs**. Specifies a list of ACIs relevant to the current evaluation. These ACIs are separated by type into "Denies" and "Allows".
- **Deny ACI Evaluations**. Specifies the results of evaluating each "deny" ACI. If any of these evaluate to TRUE, then the operation will be denied.
- **Allow ACI Evaluations**. Specifies the results of evaluating each "allow" ACI. At least one of these must evaluate to "TRUE" or the operation will be denied.

For users with the `bypass-acl` privilege, the Debug ACI Logger will not provide any ACI debug tracing since evaluations are not done for those operations. However, you will see the following trace if you have ACI debugging enabled (`debug-aci-enabled` is set to TRUE) for those operations:

```
Bypassing ACL Evaluation for Operation
```

To avoid unnecessary tracing of these operations, the "Debug ACI Logger" uses a "Client Connection Criteria" called "Clients subject to Access Control" that excludes requests from users with the `bypass-acl` privilege. It is recommended that you create and use your own criteria which specifically targets the clients that you are trying to debug in order to make analyzing the tracing output easier.

```
$ bin/dsconfig create-connection-criteria \
  --criteria-name "Restricted Clients" \
  --type simple \
  --set none-included-user-privilege:bypass-acl
```

ⓘ **Note:** Do not use Result Criteria with the Debug ACI Logger. Result criteria is evaluated after ACIs, so it will not be taken into consideration for this type of debugging.

Problems with the Administrative Console

If a problem arises when trying to use the Administrative Console, then potential reasons for the problem may include the following:

- The web application container used to host the console is not running. If an error occurs while trying to start it, then consult the logs for the web application container.
- If a problem occurs while trying to authenticate to the web application container, then make sure that the target Directory Server is online. If it is online, then the access log may provide information about the reasons for the authentication failure.
- If a problem occurs while attempting to interact with the Directory Proxy Server instance using the Administrative Console, then the access and error logs for that Directory Server instance might provide additional information about the underlying problem.

Problems with the Administrative Console: JVM memory issues

Console runs out of memory (PermGen). An inadequate PermSize setting in the server, while hosting web applications like the Administrative Console may result in errors like this in the error log:

```
[02/Mar/2016:07:50:27.017 -0600] threadID=2 category=UTIL
    severity=SEVERE_ERROR msgID=-1 msg="The server experienced an unexpected
    error. Please report this problem and include this log file.
    OutOfMemoryError: PermGen space
    ()\ncom.unboundid.directory.server.core.DirectoryServer.uncaughtException
    (DirectoryServer.java:15783)\njava.lang.ThreadGroup.uncaughtException
    (ThreadGroup.java:1057)\njava.lang.ThreadGroup.uncaughtException
    (ThreadGroup.java:1052)\njava.lang.ThreadGroup.uncaughtException
    (ThreadGroup.java:1052)\njava.lang.Thread.dispatchUncaughtException
    (Thread.java:1986)\nBuild revision: 22496\n"
```

This is only relevant for servers running Java 7.

Problems with the HTTP Connection Handler

When problems with the HTTP Connection Handler occur, first look at the HTTP connection handler log to diagnose the issue. The following section shows HTTP log examples when various errors occur.

- **Failed Request Due to a Non-Existent Resource**. The server receives a status code 404, which indicates the server could not match the URI.

  ```
  [15/Mar/2012:17:39:39 -0500] RESULT requestID=0 from="10.2.1.113:52958"
  method="GET" url="https://10.2.1.113:443/Aleph/Users/uid=user.1,ou=people,
  dc=example,dc=com" requestHeader="Host: x2270-11.example.lab"
  requestHeader="Accept: */*" requestHeader="User-Agent: curl/7.21.6
  (i386-pc-centos2.10) libcurl/7.21.6 OpenSSL/1.0.0d zlib/1.2.5 libidn/1.22
  libssh2/1.2.7" authorizationType="Basic" statusCode=404 etime=81.484
  responseContentLength=103 responseHeader="Access-Control-Allow-
  Credentials:true"
  responseContentType="application/json"
  ```

- **Failed Request due to a Malformed Request Body**. The server receives a status code 400, which indicates that the request had a malformed syntax in its request body.

  ```
  [15/Mar/2012:17:47:23-0500] RESULT requestID=10 from="10.2.1.113:55284"
  method="POST" url="https://10.2.1.113:443/Aleph/Users" requestHeader="Host:
  x2270-11.example.lab" requestHeader="Expect: 100-continue"
  requestHeader="Accept: */*" requestHeader="Content-Type: application/json"
  requestHeader="User-Agent: curl/ 7.21.6 (i386-pc-centos2.10) libcurl/7.21.6
  OpenSSL/1.0.0d zlib/1.2.5 libidn/1.22 libssh2/1.2.7"
   authorizationType="Basic"
  requestContentType="application/json" requestContentLength=5564
   statusCode=400
  etime=15.272 responseContentLength=133 responseContentType="application/
  json"
  ```

- **Failed Request due to an unsupported HTTP method**. The server receives a status code 405, which indicates that the specified method (e.g., "PATCH") in the request line is not allowed for the resource identified in the URI.

```
[15/Mar/2012:17:48:59-0500] RESULT requestID=11 from="10.2.1.113:55763"
method="PATCH" url="https://10.2.1.113:443/Aleph/Users"
 requestHeader="Host:
x2270-11.example.lab" requestHeader="Accept: */*" requestHeader="Content-
Type:
application/json" requestHeader="User-Agent: curl/7.21.6 (i386-pc-
centos2.10)
libcurl/7.21.6 OpenSSL/1.0.0d zlib/1.2.5 libidn/1.22 libssh2/1.2.7"
authorization-Type="Basic" requestContentType="application/json"
 statusCode=405
etime=6.807 responseContentLength=0 responseHeader="Allow: POST, GET,
 OPTIONS, HEAD"
```

- **Failed Request due to an Unsupported Media Type**. The server receives a status code 415, which indicates that the request entity is in a format that is not supported by the requested resource.

```
[15/Mar/2012:17:44:45-0500] RESULT requestID=4 from="10.2.1.113:54493"
method="POST" url="https://10.2.1.113:443/Aleph/Users" requestHeader="Host:
x2270-11.example.lab" requestHeader="Accept: */*" requestHeader="Content-
Type:
application/atom+xml" requestHeader="User-Agent: curl/7.21.6 (i386-pc-
centos2.10)
libcurl/7.21.6 OpenSSL/1.0.0d zlib/1.2.5 libidn/1.22 libssh2/1.2.7"
authorizationType="Basic" requestContentType="application/atom+xml"
requestContentLength=3 statusCode=415 etime=6.222
 responseContentLength=1402
responseHeader="Cache-Control: must-revalidate,no-cache,no-store"
responseContentType="text/html;charset=ISO-8859-1"
```

- **Failed Request due to an Authentication Error**. The server receives a status code 401, which indicates that the request requires user authentication.

```
[15/Mar/2012:17:46:06-0500] RESULT requestID=8 from="10.2.1.113:54899"
method="GET" url="https://10.2.1.113:443/Aleph/Schemas"
 requestHeader="Host:
x2270-11.example.lab" requestHeader="Accept: */*" requestHeader="User-
Agent:
curl/7.21.6 (i386-pc-centos2.10) libcurl/7.21.6 OpenSSL/1.0.0d zlib/1.2.5
libidn/1.22 libssh2/ 1.2.7" authorizationType="Basic" statusCode=401
etime=2.751 responseContentLength=63 responseHeader="WWW-Authenticate:
 Basic
realm=SCIM" responseHeader="Access-Control-Allow-Credentials: true"
responseContentType="application/json"
```

Virtual process size on RHEL6 Linux is much larger than the heap

Red Hat Linux introduced a change in glib 2.11 that creates larger per-thread address spaces aligned at 64MB. This change results in a virtual process size much larger than those seen in previous versions of **`glibc`**. This is not considered a bug by RedHat, as noted in *https://bugzilla.redhat.com/show_bug.cgi?id=640286*, and does not affect the physical memory needed by the server process. To see the version of **`glibc`** on your system, use the command **`yum info glibc`**.

Providing information for support cases

If a problem arises that you are unable to fully diagnose and correct on your own, then contact your authorized support provider for assistance. To ensure that the problem can be addressed as quickly as possible, be sure to provide all of the information that the support personnel may need to fully understand the underlying cause by running the **`collect-support-data`** tool, and then sending the generated zip file to your authorized support provider. It is good practice to run this tool and send the ZIP file to your authorized support provider before any corrective action has taken place.

Command-Line Tools

The PingDirectory Server provides a full suite of command-line tools necessary to administer the server. The command-line tools are available in the `bin` directory for UNIX or Linux systems and `bat` directory for Microsoft Windows systems.

Available command-line tools

Directory Server provides the following command-line tools, which you can run in interactive, noninteractive, or script mode.

Tools Help

For	Use this option	Example
Information about arguments and subcommands Usage examples	`--help`	**`dsconfig --help`**
A list of subcommands	`--help-subcommands`	**`dsconfig --help-subcommands`**
More information about a subcommand	`--help` with the subcommand	**`dsconfig list-log-publishers --help`**

ⓘ **Note:** For detailed information and examples of the command-line tools, see the *Ping Identity Directory Server Command-Line Tool Reference*.

Command-Line Tools

audit-data-security	Perform an internal task that examines all or a subset of entries in the server, writing a series of reports on potential risks with the data. Reports are written to the output directory organized by backend name and audit items.
authrate	Perform repeated authentications against Directory Server, where each authentication consists of a search to find a user followed by a bind to verify the credentials for that user.
backup	Run full or incremental backups on one or more directory server backends. This tool also supports the use of a properties file to pass predefined command-line arguments. See *Saving Options in a File* for more information.
base64	Encode raw data using the base64 algorithm or decode base64-encoded data back to its raw representation.
collect-support-data	Collect and package system information useful in troubleshooting problems. The information is packaged as a zip archive that can be sent to a technical support representative.
config-diff	Generate a summary of the configuration changes in a local or remote server instance. The tool can be used to compare configuration settings when troubleshooting issues or when verifying configuration settings on new servers.

create-rc-script	Create a Run Control (RC) script that you can use to start, stop, and restart the Directory Server on UNIX-based systems.
create-systemd-script	Create a systemd script to start and stop the Directory Server on Linux-based systems.
dbtest	Inspect the contents of Directory Server local DB backends that store their information in Berkeley DB Java Edition databases. Only backends of type local DB can be inspected by this tool.
deliver-one-time-password	Submit a "deliver one-time password" extended request, OID 1.3.6.1.4.1.30221.2.6.24, to the server which results in a the generation of a one-time password that is delivered out-of-band to the specified user. This tool can be used to test the UNBOUNDID-DELIVERED-OTP SASL mechanism.
deliver-password-reset-token	Generate and deliver a single-use token to a user through some out-of-band mechanism. The user can provide that token to the password modify extended request in lieu of the user's current password in order to select a new password.
dsconfig	View and edit the Directory Server configuration.
dsjavaproperties	Configure the JVM arguments used to run the Directory Server and associated tools. Before launching the command, edit the properties file located in `config/java.properties` to specify the desired JVM options and JAVA_HOME environment variable.
dsreplication	Manage data replication between two or more Directory Server instances.
dump-dns	Obtain a listing of all of the DNs for all entries below a specified base DN in the Directory Server.
encode-password	Encode user passwords with a specified storage scheme or determine whether a given clear-text value matches a provided encoded password.
encrypt-file	Encrypt or decrypt data using a key generated from a user-supplied passphrase, a key generated from an encryption settings definition, or a key shared among servers in the topology. The data to be processed can be read from a file or standard input, and the resulting data can be written to a file or standard output. You can use this command to encrypt and subsequently decrypt arbitrary data, or to decrypt encrypted backups, LDIF exports, and log files generated by the server.
encryption-settings	Manage the server encryption settings database.
enter-lockdown-mode	Request that the Directory Server enter lockdown mode, during which it only processes operations requested by users holding the `lockdown-mode` privilege.
export-ldif	Export data from the Directory Server backend in LDIF form.
export-reversible-passwords	Requests that the server export entries from a specified backend in LDIF form, including clear-text representations of any passwords encoded with a reversible storage scheme. This tool can only be used over a secure connection and when authenticated as a user with the permit-export-reversible-passwords privilege. The output is encrypted using a key generated from either a user-supplied passphrase or an encryption settings definition.

extract-data-recovery-log-changes	Extract changes matching a given set of criteria from a Directory Server audit log so that they can be replayed (for example, as part of a disaster recovery process) or reverted (for example, to back out changes made in error).
generate-totp-shared-secret	Generate a shared secret that you can use to generate time-based one-time password (TOTP) authentication codes for use in authenticating with the UNBOUNDID-TOTP SASL mechanism or with the validate TOTP password extended operation.
identify-references-to-missing-entries	Identify entries containing one or more attributes that reference entries that do not exist. This might require the ability to perform unindexed searches and/or the ability to use the simple paged results control.
identify-unique-attribute-conflicts	Identify unique attribute conflicts. The tool can identify values of one or more attributes that are supposed to exist only in a single entry but are found in multiple entries.
import-ldif	Import LDIF data into the Directory Server backend.
indent-ldap-filter	Parse a provided LDAP filter string and display it a multiline form that makes it easier to understand its hierarchy and embedded components. If possible, it might also simplify the provided filter in certain ways (for example, by removing unnecessary levels of hierarchy, like an AND embedded in an AND).
ldap-debugger	Intercept and decode LDAP communication.
ldap-diff	Compare the contents of two LDAP servers.
ldap-result-code	Display and query LDAP result codes.
ldapcompare	Perform compare operations in the Directory Server. Compare operations can be used to efficiently determine whether a specified entry has a given value.
ldapdelete	Delete one or more entries from an LDAP directory server. You can provide the DNs of the entries to delete using named arguments, as trailing arguments, from a file, or from standard input. Alternatively, you can identify entries to delete using a search base DN and filter.
ldapmodify	Apply a set of add, delete, modify, and/or modify DN operations to a directory server. Supply the changes to apply in LDIF format, either from standard input or from a file specified with the `ldifFile` argument. Change records must be separated by at least one blank line.
ldappasswordmodify	Update the password for a user in an LDAP directory server using the password modify extended operation (as defined in RFC 3062), a standard LDAP modify operation, or an Active Directory-specific modification.
ldapsearch	Process one or more searches in the Directory Server.
ldif-diff	Compare the contents of two files containing LDIF entries. The output is an LDIF file containing the add, delete, and modify change records needed to convert the data in the source LDIF file into the data in the target LDIF file.
ldifmodify	Apply a set of changes (including add, delete, modify, and modify DN operations) to a set of entries contained in an LDIF file. The changes are read from a second file (containing change records rather than entries), and the updated entries are written to a third LDIF file. Unlike ldapmodify, ldifmodify cannot read the changes to apply from standard input.

ldifsearch	Search one or more LDIF files to identify entries matching a given set of criteria.
leave-lockdown-mode	Request that the Directory Server leave lockdown mode and resume normal operation.
list-backends	List the backends and base DNs configured in Ping Identity Directory Server.
load-ldap-schema-file	Load the schema definitions contained in a specified LDIF file into the schema for a running server. You can only use this command with a server instance running on the local system.
make-ldif	Generate LDIF data based on a definition in a template file. For example template files, see the server's `config/MakeLDIF` directory. In particular, the `examples-of-all-tags.template` file shows how to use all of the tags for generating values.
manage-account	Retrieve or update information about the current state of a user account. Processing is performed using the password policy state extended operation, and you must have the `password-reset` privilege to use this extended operation.
manage-certificates	Manage certificates and private keys in a JKS or PKCS #12 key store.
manage-extension	Install or update extension bundles. An extension bundle is a package of extensions that use the Server SDK to extend the functionality of the PingDirectory Server. Extension bundles are installed from a zip archive or file system directory. PingDirectory Server will be restarted if running to activate the extensions.
manage-profile	Generate, compare, install, and replace server profiles.
manage-tasks	Access information about pending, running, and completed tasks scheduled in the Directory Server.
manage-topology	Manage the topology registry.
migrate-ldap-schema	Migrate schema information from an existing LDAP server into a Ping Identity Directory Server instance.
migrate-sun-ds-config	Update an instance of the Ping Identity Directory Server to match the configuration of an existing Sun Java System Directory Server 5.x, 6.x, or 7.x.
modrate	Perform repeated modifications against an LDAP directory server.
move-subtree	Move all entries in a specified subtree from one server to another.
parallel-update	Perform add, delete, modify, and modify DN operations concurrently using multiple threads.
populate-composed-attribute-values	Populate entries in one or more backends with attribute values generated by one or more composed attribute plugins.
profile-viewer	View information in data files captured by the Directory Server profiler.
re-encode-entries	Initiate a task that causes a local DB backend to re-encode all or a specified subset of the entries that it contains. The tool does not alter the entries themselves but provides a useful mechanism for applying significant changes to the way that entries are stored in the backend (for example, to apply encoding changes if a feature like data encryption or uncached attributes or entries is enabled).

rebuild-index	Rebuild index data within a backend based on the Berkeley DB Java Edition. Note that this tool uses different approaches to rebuilding indexes based on whether it is running in online mode (as a task) rather than with the server offline. Running in offline mode will often provide significantly better performance and require significantly less database cleaning, particularly for indexes containing keys that match a large number of entries and have high index entry limit and exploded index entry threshold values. Also note that rebuilding an index with the server online will prevent the server from using that index while the rebuild is in progress, so some searches might behave differently while a rebuild is active than when it is not.
register-yubikey-otp-device	Register a YubiKey OTP device with the Directory Server for a specified user so that the device can be used to authenticate that user in conjunction with the UNBOUNDID-YUBIKEY-OTP SASL mechanism. Alternately, it can be used to deregister one or more YubiKey OTP devices for a user so that they can no longer be used to authenticate that user.
reload-http-connection-handler-certificates	Reload HTTPS Connection Handler certificates.
remove-attribute-type-from-schema	Safely remove an attribute type definition from the server schema.
remove-backup	Safely remove a backup and optionally all of its dependent backups from the specified Directory Server backend.
remove-defunct-server	Remove a server from this server's topology.
replace-certificate	Replace the listener certificate for this Ping Identity Directory Server server instance.
restore	Restore a backup of a Directory Server backend.
revert-update	Revert this server package's most recent update.
review-license	Review and/or indicate your acceptance of the license agreement defined in `legal/LICENSE.txt`.
rotate-log	Trigger the rotation of one or more log files.
sanitize-log	Sanitize the contents of a server log file to remove potentially sensitive information while still attempting to retain enough information to make it useful for diagnosing problems or understanding load patterns. The sanitization process operates on fields that consist of name-value pairs. The field name is always preserved, but field values might be tokenized or redacted if they might include sensitive information. Supported log file types include the file-based access, error, sync, and resync logs, as well as the operation timing access log and the detailed HTTP operation log. Sanitize the audit log using the scramble-ldif tool.

schedule-exec-task	Schedule an exec task to run a specified command in the server. To run an exec task, a number of conditions must be satisfied: the server's global configuration must have been updated to include 'com.unboundid.directory.server.tasks.ExecTask' in the set of allowed-task values, the requester must have the `exec-task` privilege, and the command to execute must be listed in the `exec-command-whitelist.txt` file in the server's `config` directory. The absolute path (on the server system) of the command to execute must be specified as the first unnamed trailing argument to this program, and the arguments to provide to that command (if any) should be specified as the remaining trailing arguments. The server root is used as the command's working directory, so any arguments that represent relative paths are interpreted as relative to that directory.
search-and-mod-rate	Perform repeated searches against an LDAP directory server and modify each entry returned.
search-logs	Search across log files to extract lines matching the provided patterns, like the **`grep`** command-line tool. The benefits of using this tool over **`grep`** are its ability to handle multiline log messages, extract log messages within a given time range, and the inclusion of rotated log files.
searchrate	Perform repeated searches against an LDAP directory server.
server-state	View information about the current state of the Directory Server process.
set-delegated-admin-aci	Request that the Directory Server assign appropriate ACI for configured delegated administrators of the Delegated Admin API.
setup	Perform the initial setup for a server instance.
start-server	Start the Directory Server.
status	Display basic server information.
stop-server	Stop or restart the server.
subtree-accessibility	List or update the set of subtree accessibility restrictions defined in the Directory Server.
sum-file-sizes	Calculate the sum of the sizes for a set of files.
summarize-access-log	Generate a summary of one or more access logs to display a number of metrics about operations processed within the server.
transform-ldif	Apply one or more changes to entries or change records read from an LDIF file, writing the updating records to a new file. This tool can apply a variety of transformations, including scrambling attribute values, redacting attribute values, excluding attributes or entries, replacing existing attributes, adding new attributes, renaming attributes, and moving entries from one subtree to another.
uninstall	Uninstall Ping Identity Directory Server.

update	Update the Directory Server to a newer version by downloading and unzipping the new server install package on the same host as the server you wish to update. Then, use the **`update`** tool from the new server package to update the older version of the server. Before upgrading a server, you should ensure that it is capable of starting without severe or fatal errors. During the update process, the server is stopped if running, then the update is performed, and a check is made to determine if the newly updated server starts without major errors. If it cannot start cleanly, the update will be backed out and the server returned to its prior state. See the **`revert-update`** tool for information on reverting an update.
validate-acis	Validate a set of access control definitions contained in an LDAP server (including Sun/Oracle DSEE instances) or an LDIF file to determine whether they are acceptable for use in the Directory Server. Note that the output generated by this tool will be in LDIF format, but each entry in the output will have exactly one ACI, so entries that have more than one ACI will appear multiple times in the output with different ACI values.
validate-file-signature	Validate file signatures. For best results, file signatures should be validated by the same instance used to generate the file. However, it might be possible to validate signatures generated on other instances in a replicated topology.
validate-ldap-schema	Validate an LDAP schema read from one or more LDIF files.
validate-ldif	Validate the contents of an LDIF file against the server schema.
verify-index	Verify that indexes in a backend using the Berkeley DB Java Edition are consistent with the entry data contained in the database.
watch-entry	Launch a window to watch an LDAP entry for changes. If the entry changes, the background of modified attributes will temporarily be red. Attributes can be modified as well. This tool is primarily intended to demonstrate replication or synchronization functionality.

Saving options in a file

Directory Proxy Server supports the use of a tools properties file (`config/tools.properties`) to simplify command-line invocations by reading in a set of options for each tool from a text file.

Properties files are convenient when quickly testing Directory Proxy Server in multiple environments.

Each property takes the form of a name-value pair that defines predetermined values for a tool's options.

Directory Proxy Server supports the following types of properties:

- Default properties that apply to all command-line utilities
- Tool-specific properties

Creating a tools properties file

You can set properties that apply to all tools or are tool-specific. These properties serve as defaults for the command-line options they represent.

Steps

1. Use a text editor to open the default tools properties file (`config/tools.properties`) or a different properties file.

   > **Note:**
   >
   > If you use a file other than `config/tools.properties`, invoke the tool with the `--propertiesFilePath` option to specify the path to your properties file.

2. Set or change properties that apply to all tools.

   Use the standard Java properties file format (name=value) to set properties. For example, the following properties define a set of LDAP connection parameters.

   ```
   hostname=server1.example.com
   port=1389
   bindDN=cn=Directory\ Manager
   bindPassword=secret
   baseDN=dc=example,dc=com
   ```

   > **Note:**
   >
   > Properties files do not allow quotation marks of any kind around values.
   >
   > Escape spaces and special characters.
   >
   > Whenever you specify a path, do not use `~` to refer to the home directory. The server does not expand the `~` value when read from a properties file.

3. Set or change properties that apply to specific tools.

   Tool-specific properties start with the name of the tool followed by a period. These properties take precedence over properties that apply to all tools. The following example sets two ports: one that applies to all tools (`port=1389`) and a tool-specific one that **`ldapsearch`** uses instead (`ldapsearch.port=2389`).

   ```
   hostname=server1.example.com
   port=1389
   ldapsearch.port=2389
   bindDN=cn=Directory\ Manager
   ```

4. Save your changes and close the file.

Evaluation order

Directory Server uses the following evaluation ordering to determine options for a given command-line tool:

- All options on the tool's command line take precedence over any options in any properties file.
- If you use the `--propertiesFilePath` option with no other options, Directory Server takes its options from the specified properties file.
- If you use no options on the command line including the `--propertiesFilePath` option (and `--noPropertiesFile`), Directory Server searches for the `tools.properties` file at *<server-root>*.
- If no default properties file is found and a required option is missing, the tool generates an error.

- Tool-specific properties (for example, `ldapsearch.port=3389`) take precedence over general properties (for example, `port=1389`).

Consider this example properties file that is saved as *<server-root>*`/bin/tools.properties`:

```
hostname=server1.example.com
port=1389
bindDN=cn=Directory\ Manager
bindPassword=secret
```

Directory Server locates a command-line option in a specific priority order.

1. All options presented with the tool on the command line take precedence over any options in any properties file. In the following example, the client request is run with the options specified on the command line (`--port` and `--baseDN`). The command uses the `bindDN` and `bindPassword` values specified in the properties file.

   ```
   $ bin/ldapsearch --port 2389 --baseDN ou=People,dc=example,dc=com \
     --propertiesFilePath bin/tools.properties "(objectclass=*)"
   ```
2. Next, if you specify the properties file using the `--propertiesFilePath` option and no other command-line options, Directory Server uses the specified properties file as follows:

   ```
   $ bin/ldapsearch --propertiesFilePath bin/tools.properties \
     "(objectclass=*)"
   ```
3. If no options are presented with the tool on the command line and the `--noPropertiesFile` option is not present, Directory Server attempts to locate any default `tools.properties` file in the following location:

   ```
   <server-root>/config/tools.properties
   ```

   If you move your `tools.properties` file from *<server-root>*`/bin` to the *<server-root>*`/config` directory. You can then run your tools as follows:

   ```
   $ bin/ldapsearch "(objectclass=*)"
   ```

   You can Directory Server so that it does not search for a properties file by using the `--noPropertiesFile` option. This options tells Directory Server to use only those options specified on the command line. You cannot use the `--propertiesFilePath` and `--noPropertiesFile` options together.
4. If no default `tools.properties` file is found and no options are specified with the command-line tool, then the tool generates an error for any missing arguments.

Sample dsconfig batch files

The `config/sample-dsconfig-batch-files` directory contains **`dsconfig`** batch files that you can use to configure various aspects of the server. For example, these files can enable additional security capabilities or take advantage of features that might require customization from one environment to another.

Each file includes comments that describe the purpose and benefit of its configuration change. You can choose which of the changes you want to apply.

You need to customize some of the batch files to provide values that might vary from one environment to another. To apply a batch file that requires changes, copy it to another directory and edit the copy. Leave the files in the `config/sample-dsconfig-batch-files` directory unchanged so that they can be updated when you upgrade the server. To specify the path to the file that contains the changes to apply, use the **`dsconfig`** tool (**`bin/dsconfig`** on UNIX-based systems or **`bat\dsconfig.bat`** on Windows) with the `--batch-file` argument.

You should also provide the arguments needed to connect and authenticate to the server. The `--no-prompt` argument ensures that the tool does not block while waiting for input if any necessary arguments are missing. Consider this example.

```
bin/dsconfig --hostname localhost \
  --port 636 --useSSL --trustStorePath config/truststore \
  --bindDN "uid=admin,dc=example,dc=com" \
  --bindPasswordFile admin-password.txt \
  --batch-file config/hardening-dsconfig-batch-files/reject-insecure-
request.dsconfig \
  --no-prompt
```

Running task-based tools

Directory Server has a Tasks subsystem that allows you to schedule basic operations, such as **`backup`**, **`restore`**, **`rotate-log`**, **`schedule-exec-task`**, **`stop-server`**, and others. All task-based tools require the `--task` option that explicitly indicates the tool is to run as a task rather than in offline mode.

The following table shows the options that you can use for task-based operations:

Options for task-based operations

Option	Description
`--task`	Indicates that the tool is invoked as a task. The `--task` option is required. If you invoke a tool as a task without this `--task` option, then a warning message is displayed stating that it must be used. If the `--task` option is provided but the tool was not given the appropriate set of authentication arguments to the server, then an error message is displayed and the tool exits with an error.
`--start` *<startTime>*	Indicates the date and time, expressed in the format 'YYYYMMDDhhmmss', when the operation is to start. A value of '0' causes the task to be scheduled for immediate execution. After the scheduled run, the tool exits immediately.
`--dependency` *<taskID>*	Specifies the ID of a task upon which this task depends. A task does not start execution until all its dependencies have completed execution. You can use this option multiple times in a single command.
`--failedDependencyAction` *<action>*	Specifies the action this task takes if one of its dependent tasks fail. Valid action values are: ▪ `CANCEL` (the default) Cancels the task. ▪ `DISABLE` Disables the task so that it is not eligible to run until you manually enable it again. ▪ `PROCESS` Runs the task.

Option	Description
`--startAlert`	Generates an administrative alert when the task starts running.
`--errorAlert`	Generates an administrative alert when the task fails to complete successfully.
`--successAlert`	Generates an administrative alert when the task completes successfully.
`--startNotify` *`<emailAddress>`*	Specifies an email address to notify when the task starts running. You can use this option multiple times in a single command.
`--completionNotify` *`<emailAddress>`*	Specifies an email address to notify when the task completes, regardless of whether it succeeded or failed. You can use this option multiple times in a single command.
`--errorNotify` *`<emailAddress>`*	Specifies an email address to notify if an error occurs when this task executes. You can use this option multiple times in a single command.
`--successNotify` *`<emailAddress>`*	Specifies an email address to notify when this task completes successfully. You can use this option multiple times in a single command.

PingDirectoryProxy Server Administration Guide

PingDirectory™ Product Documentation

Support

https://support.pingidentity.com/

Introduction

PingDirectoryProxy™ Server is a fast and scalable LDAPv3 gateway for the PingDirectory® Server. The Directory Proxy Server architecture can be configured to control how client requests are routed to backend servers.

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

Overview of the PingDirectoryProxy Server features

The PingDirectoryProxy Server is a fast, scalable, and easy-to-use LDAP proxy server that provides high availability and additional security for the PingDirectory Server, while remaining largely invisible to client applications. From a client perspective, request processing is the same, whether communicating with the Directory Server directly or going through the Directory Proxy Server.

The PingDirectoryProxy Server provides the following set of features:

- **High availability**. The Directory Proxy Server allows you to transparently fail over between servers if a problem occurs, as well as ensuring that the workload is balanced across the topology. If a client does not support following referrals, the Directory Proxy Server can follow them on the client's behalf.
- **Data mapping and transformation**. The Directory Proxy Server can do DN mapping and attribute mapping to allow clients to interact with the server using older names for directory content. It allows clients to continue working when they would not be able to work directly with the Directory Server, either because of changes that have occurred at the data layer or to inherent design limitations in the clients.
- **Horizontal scalability and performance**. Reads can be horizontally scaled using load balancing. In large data centers, if the data set is too large to be cached or to provide horizontal scalability for writes, the Directory Proxy Server can automatically split the data across multiple systems. This feature allows the Directory Proxy Server to improve scalability and performance of the Directory Server environment.
- **Load balancing and failover**. You can spread the workload across multiple proxies in a large data center using load-balancing algorithms. Load balancing is also useful when a server becomes degraded or non-responsive, because client process requesting is directed to a different server.
- **Security and access control**. The Directory Proxy Server can add additional firewall capabilities, as well as constraints and filtering to help protect the Directory Server from attacks. You can use a Directory Proxy Server in a DMZ as opposed to allowing clients to directly access the Directory Proxy Server in the internal network or providing the data in the DMZ. It can help provide secure access to the data and you can define what actions clients are allowed to do. For example, you can prevent clients from making modifications to data when connected via a VPN no matter what their identity or permissions.
- **Tracking of operations across the environment**. In the past, administrators have commonly complained that when they see a request in the access log, they have no idea where it came from and cannot track it back to a particular client. The Directory Proxy Server contains controls that allow administrators to track requests back to the client that issued them. Whenever the Directory Proxy Server forwards a request to the Directory Server, it includes a control in the request so that the Directory Server's access log has the IP address of the client, address and connection ID of the Directory Server. In the response back to the client, it similarly includes information about the Directory Server that processed the request, such as the connection ID and operation ID. This feature makes it easier for administrators to keep track of what is going on in their environment.
- **Monitoring and management tools**. Because the Directory Proxy Server uses many of the components of the PingDirectory Server, it can leverage them to provide protocol support, logging, management tools for configuration and monitoring, schema, and so on. You can use the Data Metrics Server, the **`dsconfig`** tool and the Administrative Console to manage the Directory Proxy Server.

Overview of the Directory Proxy Server components and terminology

The Directory Proxy Server consists of the following components and functionality that provide the proxy capabilities:

- Locations
- LDAP External Servers
- LDAP Health Checks
- Load-Balancing Algorithms
- Data Transformations
- Request Processors
- Server Affinity Providers
- Subtree Views
- Connection Pools
- Client Connection Policies
- Entry Balancing

This section describes each component in more detail.

About locations

Locations define a group of servers with similar response time characteristics. Each location consists of a name and an ordered list of preferred failover locations. The Directory Proxy Server and each of the backend LDAP external servers can be assigned locations. These locations can be taken into account when deciding how to route requests, so that the server prefers to forward requests to Directory Server in the same data center over those in remote locations. As a rule of thumb, if you have multiple data centers then you should have a separate location for each one. In most environments, all Directory Proxy Server instances should have the same configuration except for the attribute that specifies the location of the Directory Proxy Server itself.

For example, a deployment consists of three data centers, one in New York, another in Chicago, and another in Los Angeles. In the New York data center, applications which reside in this data center prefer communicating with directories in this data center. If none of the servers are available, it prefers to failover to the data center in Chicago rather than the data center in Los Angeles. So the New York location contains an ordered list in which the Chicago location is preferred over the Los Angeles data center for failover.

For information about configuring locations, see *Configuring Locations*.

About LDAP external servers

You can configure information about the directory server instances accessed by the PingDirectoryProxy Server. This configuration information includes the following:

- Server connection information, such as IP address, port, and security layer
- Location
- Authentication information
- Methods for authenticating and authorizing clients
- Server-specific health checks
- Types of operations allowed. For example, some LDAP external servers may allow only reads and others allow reads and writes, so the Directory Proxy Server can recognize this and accommodate it.

The PingDirectoryProxy Server allows you to configure different types of LDAP external servers. The default configuration for each type is tuned to be the best possible configuration for each.

For information about configuring LDAP external servers, see *Configuring LDAP External Servers*.

About LDAP health checks

The LDAP health check component provides information about the availability of LDAP external servers. The health check result includes a server state, which can be one of the following:

- **Available**. Completely accessible for use.
- **Degraded**. The server may be used if necessary, but has a condition which may make it less desirable than other servers (for example, it is slow to respond or has fallen behind in replication).

- **Unavailable**. Completely unsuitable for use (for example, the server is offline or is missing critical data).

Health check results also include a numeric score, which has a value between 1 and 10, that can help rank servers with the same state. For example, if two servers are available and one has a score of 8 and the other a score of 7, the Directory Proxy Server can be configured to prefer the server with the higher score.

The Directory Proxy Server periodically invokes health checks to monitor each LDAP external server, and may also initiate health checks in response to failed operations. It checks the health of the LDAP external servers at intervals configured in the LDAP server's `health-check-frequency` property. However, the Directory Proxy Server has safeguards in place to ensure that only one health check is in progress at any time against a backend server to avoid affecting its ability to process other requests.

The results of health checks performed by the Directory Proxy Server are made available to the load-balancing algorithms so that they may be taken into account when determining where to send requests. The Directory Proxy Server will attempt to use servers with a state of available before trying servers with a state of degraded. It will never attempt to use servers with a state of unavailable. Some load-balancing algorithms may also take the health check score into account, such as the health-weighted load-balancing algorithm, which prefers servers with higher scores over those with lower scores. You configure the algorithms that work best for you environment.

In some cases, an LDAP health check may define different sets of criteria for promoting and demoting the state of a server. So, a degraded server may need to meet more stringent require- ments to be reclassified as available than it originally took to be considered degraded. For example, if response time is used in the process of determining the health of a server, then the Directory Proxy Server may have a faster response time threshold for transitioning a server from degraded back to available than the threshold used to consider it degraded in the first place. This threshold difference can help avoid cases in which a server repeatedly transitions between the two states because it is operating near the threshold.

For example, the health check used to measure search response time is configured to mark any server to be marked degraded when the search response time is greater than 1 second. You can then configure that the response time must be less than 500 ms before the server is made available again, so that the Directory Proxy Server does not flip back and forth between available and degraded.

PingDirectoryProxy Server provides the following health checks:

- **Measure the response time for searches and examine the entry contents**. For example, the health check might retrieve a monitoring entry from a server and base the health check result on whether the entry was returned, how long it took to be returned, and whether the value of the returned entry matches what was expected.
- **Monitor the replication backlog**. If a server falls too far behind in replication, then the Directory Proxy Server can stop sending requests to it. A server is classified as degraded or unavailable if the threshold is reached for the number of missing changes, the age of the oldest missing change, or both.
- **Consume Directory Server administrative alerts**. If the Directory Server indicates there is a problem, for example an index that must be rebuilt, then it will flag itself as degraded or unavailable. When the Directory Proxy Server detects this, it will stop sending requests to the server. The Directory Proxy Server detects administrative alerts as soon as they are issued by maintaining an LDAP persistent search for changes within the `cn=alerts` branch of the Directory Server. When the Directory Proxy Server is notified by the Directory Server of a new alert, it immediately retrieves the base `cn=monitor` entry of the Directory Server. If this entry has a value for the `unavailable-alert-type` attribute, then the Directory Proxy Server will consider it unavailable. If this entry has a value for the `degraded-alert-type` attribute, then the Directory Proxy Server will consider it degraded. Clients of the Directory Proxy Server can use a similar mechanism to detect and react when a Directory Proxy Server flags itself as degraded or unavailable.
- **Monitor the busyness of the server**. If a server becomes too busy, then it may be marked degraded or unavailable so that less heavily-loaded servers may be preferred.

For information about configuring health checks, see *Configuring Server Health Checks*. To associate a health check with an LDAP external server and set the health check frequency, you must configure the `health-check` and `health-check-frequency` properties of the LDAP external server. See

“To Configure an External Server Using dsconfig” for information about configuring the properties of the external server.

About load-balancing algorithms

Load-balancing algorithms are used to determine which server in a set of similar servers should be used to process a client request. The algorithm can take the following criteria into account:

- **Consider the location of the server**. Servers in the same location as the Directory Proxy Server can be preferred over those in alternate locations.
- **Consider the health of the server**. Servers that are available are preferred over those that are degraded. In some cases, the health check score may also be used to further differentiate between servers with the same health check state.
- **Route requests consistently**. Requests from a single client may be consistently routed to the same directory server instance to avoid problems such as propagation delay from replication.
- **Retry the operation in an alternate server if the request fails or the operation times out**. You can control if the retry is allowed and, if so, how many times to retry and the time out interval.

The PingDirectoryProxy Server provides the following load-balancing algorithms:

- **Fewest operations**. Requests are forwarded to the backend server with the fewest operations currently in progress.
- **Single server**. Requests are always sent to the same server and will not attempt to fail over to another server if the target server is unavailable.
- **Weighted**. Administrators explicitly assign numeric weights to individual servers or sets of servers to control how likely they are to be selected for processing requests relative to other servers.
- **Health-based weighting**. Uses the health check score to assign weights to each of the servers, so that a server with a higher score gets a higher percentage of the traffic than a server with a lower score. The proportion of traffic received is the difference between their health check scores.
- **Failover**. Requests are always sent to a given server first. If that server fails, then the request is sent to another specified server, and so on through an ordered failover server list.

For information about configuring load balancing, see *Configuring Load Balancing*.

About proxy transformations

Proxy transformations are used to rewrite requests and responses as they pass through the Directory Proxy Server. Proxy data transformations are helpful for clients that use an old schema or that contain a hard-coded schema.

Proxy transformations can provide DN and attribute mapping altering both requests to the server as well as responses from the server. For example, a client sends a request to `o=example.com` even though the directory server handling the request uses `dc=example,dc=com`. The Directory Proxy Server can transparently remap the request so that the server can process it, and map it back to the original DN of the client request when the value is returned. Or if a client tries to use the attribute `userID`, the Directory Proxy Server can map it to `uid` before sending the request on to the backend LDAP server. The Directory Proxy Server then remaps the response to `userID` when the value is returned.

The Directory Proxy Server also includes a proxy transformation that can be used to suppress a specified attribute, so that it will never be returned to clients. It can also cause the server to reject requests which target that particular attribute. Another proxy transformation can be used to prevent entries that match a given search filter from being returned to clients.

For information about configuring proxy transformations, see *Configuring Proxy Transformations* on page 70.

About request processors

A request processor encapsulates the logic for handling an operation, ensuring that a given operation is handled appropriately. The request processor can either process the operation directly, forward the request to another server, or hand off the request to another request processor.

PingDirectoryProxy Server provides the following types of request processor:

- **Proxying request processors**, which forward operations received by the Directory Proxy Server to other LDAP external servers.
- **Entry-balancing request processors**, which split data across multiple servers. They determine which set of servers are used to process a given operation. They then hand off operations to proxying request processors so that requests can be forwarded to one of the servers in the set.
- **Failover request processors**, which perform ordered failover between other types of request processors, sometimes with different behavior for different types of operations.

Directory Proxy Server request processors can be used to forward certain controls, including the batch transaction control and the LDAP join control. The batch transaction control must target a single Berkley DB backend. For more information about the controls, refer to the LDAP SDK for Java documentation.

For information about configuring request processors, see *Configuring Request Processors* on page 72.

About server affinity providers

The server affinity provider can be used to establish an affinity to a particular backend server for certain operations. You can configure one of three types of provider:

- **Client connection Server Affinity**, so that requests from the same client connection may consistently be routed to the same backend server.
- **Client IP address Server Affinity**, so that all requests coming from the same client system will be consistently routed to the same backend server.
- **Bind DN Server Affinity**, so that all requests from the same user will be consistently routed to the same backend server.

For information about configuring server affinity, see *Configuring Server Affinity*.

About subtree views

A subtree view can be used to make a portion of the DIT available to a client by associating a request processor with a base DN. Subtree views allow you to route operations concerning one set of data to a particular set of data sources, and operations concerning another set of data to another set of data sources. Multiple subtree views may be involved in processing a request, such as for searches that have a scope that is larger than the subtree view.

The subtree view includes a single base DN used to identify the portion of the DIT. They may have hierarchical relationships, for example one subtree view could be configured for `dc=example,dc=com` and another for `ou=People,dc=example,dc=com`.

For information about configuring a subtree view, see *Configuring Subtree Views*.

About the connection pools

Based on the type of backend server that you are using, the PingDirectoryProxy Server maintains either one or two connection pools to the backend server. It maintains either one pool for all types of operations or two separate pools for processing bind and non-bind operations from clients. When the Directory Proxy Server establishes connections, it authenticates them using whatever authentication mechanism is defined in the configuration of the external server. These connections will be re-used for all types of operations to be forwarded to the backend server. The bind DN and password are configured in the Directory Proxy Server.

Whenever a client sends a bind request to the Directory Proxy Server, the server looks at the type of bind request that was sent. If it is a SASL bind request, then the authentication is processed by the Directory Proxy Server itself and it will not be forwarded to the backend server. However, the Directory Proxy Server may use information contained in the backend server as needed. If the bind request is a simple bind request and the bind DN is within the scope of data supplied by the backend server, then the Directory Proxy Server will forward the client request to the backend server so that it will use the credentials provided by the client.

Regardless of the authentication method that the client uses, the Directory Proxy Server will remember the identity of the client after the authentication is complete and for any subsequent requests sent by that client, it will use the configured authorization method to identify the client to the backend server. Even though the operation is forwarded over a connection that is authenticated as a user defined in the Directory Proxy Server configuration, the request is processed by the backend server under the authority of the end client.

About client connection policies

Client connection policies define the general behavior the server exhibits when communicating with a set of clients. Each policy consists of the following:

- A **set of connection criteria** that define which client is associated with the policy based on information the server has about the client, including client address, protocol used, secure communication mechanism, location of the client's entry in the Directory Server and the contents of the client's entry. These criteria are the same as those used for filtered logging. For example, different client connection policies could be established for different classes of users, such as root and non-root users.
- A **set of constraints** on the type of operations a client may request. You can specify whether a particular type of operation is allowed for clients. For some operation types, such as extended operations, you can allow only a particular subset of an operation type, such as a particular extended operation.
- A **set of subtree view**s that define information about the parts of the DIT the client may access.

When a client connection is established, only one client connection policy is applied. If the criteria for several policies match the same client connection, the evaluation order index is used as a tiebreaker. If no policy matches, the client connection is terminated. If the client binds, changing its identity, or uses StartTLS to convert from an insecure connection to a secure connection, then the connection may be evaluated again to determine if it matches the same or a different client connection policy. The connection can also be terminated if it no longer matches any policy.

For information about configuring a client connection policy, see *Configuring Client Connection Policies* on page 77.

About entry balancing

Entry balancing allows you to automatically spread entries below a common parent among multiple sets of directory servers for improved scalability and performance. Entry balancing can take advantage of a global index, an in-memory cache used to quickly determine which set or sets of servers should be used to process a request based on the entry DNs and/or the attribute values used in the request.

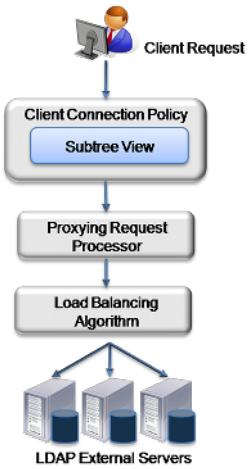
For information about configuring entry balancing, see *Deploying an entry-balancing proxy configuration* on page 1086.

Server component architecture

This section provides an overview of the process flow between the Directory Proxy Server components, for both a simple proxy deployment and an entry-balancing deployment.

Architecture of a simple Directory Proxy Server deployment

In a simple Directory Proxy Server deployment, a client request is first processed by a client connection policy as illustrated in Figure 1, “Process Flow for Directory Proxy Server”.

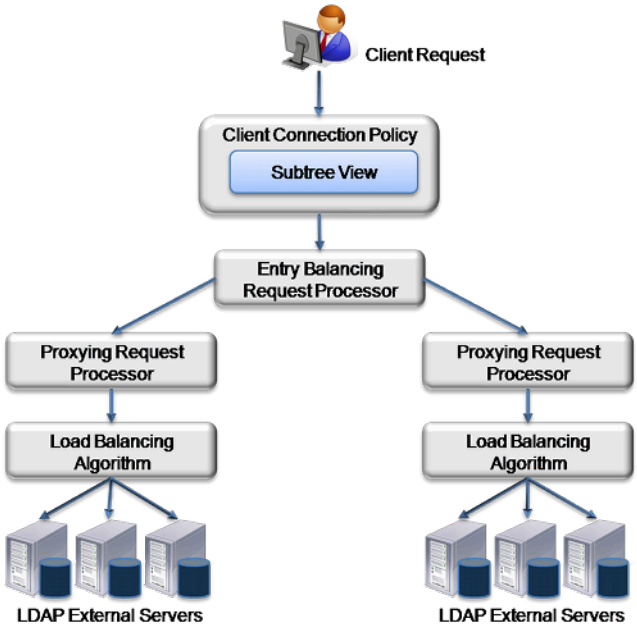


MY TITLE Process Flow for Directory Proxy Server

The client connection policy contains a subtree view, which defines the portion of the DIT available to clients. Once the Directory Proxy Server determines that the DIT is available, it passes the request to the request processor, which defines the logic for processing the request. The request processor then passes the request to a load-balancing algorithm, which determines the server in a set of servers responsible for handling the request. Finally, the request is passed to the LDAP external server. The LDAP external server contains properties that define the server's location in a topology and the health checks used to determine if the server is functioning properly. This information may be used by the load-balancing algorithm in the course of determining how to route requests.

Architecture of an entry-balancing Directory Proxy Server deployment

Figure 2, "Process Flow for Entry-Balancing Directory Proxy Server" describes how a client request is treated in an entry-balancing deployment.



MY TITLE Process Flow for Entry-Balancing Directory Proxy Server

Entry balancing is typically used when the data set it too large to fully cache on a single server or when the write performance requirements of an environment are higher than can be achieved with a single replicated

set of servers. In such cases, the data may be split across multiple sets of servers, increasing the memory available for caching and the overall write performance in proportion to the number of server sets.

As with a simple proxy deployment, the client request is first processed by the client connection policy, which determines how the Directory Proxy Server communicates with a set of clients. It contains a subtree view that represents the base DN for the entire deployment. The data set splits beneath this base DN.

The request is then passed to the entry-balancing request processor. The entry-balancing request processor contains a global attribute index property, which helps the request processor determine which server set contains the entry and how to properly route the request. It also contains a placement algorithm, which helps it select the server set in which to place new entries created by add requests.

Beneath the entry-balancing request processor are multiple proxying request processors that handle multiple unique sets of data. These request processors pass the request to a load-balancing algorithm, which determines which LDAP external server should handle the request. As with a simple proxy deployment, this LDAP external server contains properties that define the server's location and the health checks used to determine if the server is functioning properly.

For information about entry-balancing replication, see *Overview of replication in an entry-balancing environment* on page 1107.

Directory Proxy Server configuration overview

The configuration of the Directory Proxy Server involves the following steps:

- **Configuring the locations for your deployment**. A location is a collection of servers that share access and latency characteristics. For example, your deployment might include two data centers, one in the east and one in the west. These data centers would be configured as two locations in the Directory Proxy Server. Each location is associated with a name and an ordered list of failover locations, which could be used if none of the servers in the preferred location are available.
- **Configuring the Directory Proxy Server location**. You need to update the configuration to specify the location of the Directory Proxy Server instance.
- **Configuring health checks for the LDAP external servers**. You can configure at what point the Directory Proxy Server considers an LDAP external server to be available, of degraded availability, or unavailable. Each health check can be configured to be used automatically for all LDAP external servers or for a specified set of servers.
- **Configuring the LDAP external servers**. During this step, you define each of the external directory servers, including the server type. You can configure Ping Identity Directory Servers, Sun Java System Directory Servers, or generic LDAP servers. You also assign the server-specific health checks configured in the previous step.
- **Configuring the load-balancing algorithm**. You configure the load-balancing algorithm used by the Directory Proxy Server to determine which server in a set of similar servers should be used to process a client request. The Directory Proxy Server provides default algorithms. It also steps you through the creation of new algorithms by using an existing algorithm as a template or by creating one from scratch.
- **Configuring the proxying request processor**. In this step, you configure proxying request processors that forward operations received by the Directory Proxy Server to other LDAP external servers.
- **Configuring subtree views**. A subtree view defines the portion of the DIT available to a client. Each subtree view can be associated with a load-balancing algorithm to help distribute the work load.
- **Configuring the client connection policy**. You configure policies to classify how different client connections are managed by the Directory Proxy Server. The client connection policy can be used to control the types of operations that a client may perform and the portion of the DIT that the client can access. Restrictions configured in a client connection policy will take precedence over any capabilities granted by access control or privileges.

Installing the Directory Proxy Server

This section describes how to install PingDirectoryProxy Server. It includes pre-installation requirements and considerations.

It includes the following sections:

Before you begin

The following sections describe requirements and considerations you should make before installing the software and configuring the PingDirectoryProxy Server objects.

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

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

System requirements

The following section details system requirements that are qualified and certified to be compatible with the PingDirectoryProxy Server.

Differences in operating system versions, service packs, and other platform variations are supported until the platform or other required software are suspected of causing issues.

Platforms

The following operating systems support a PingDirectory Server installation and deployment.

- Windows Server 2019
- Windows Server 2016
- Red Hat Enterprise Linux ES 8.1
- Red Hat Enterprise Linux ES 7.7
- CentOS 8.1
- CentOS 7.7
- SUSE Linux Enterprise 15 SP1
- SUSE Linux Enterprise 12 SP5
- Ubuntu 18.04 LTS
- Ubuntu 16.04 LTS
- Amazon Linux 2

> ⓘ **Attention:**
>
> This product has been tested with the default configurations of all operating system components. Customized implementations or installed third-party plugins from your organization might affect the deployment of this product.

Docker

The following Docker version and operating systems are a pre-installation requirement.

- Version: Docker 19.03.5
- Host operating system: Ubuntu 18.04 LTS

- Base image operating system: Ubuntu 18.04 LTS
- Base image operating system: Alpine Linux 3.11

For more information, view the Directory Proxy Server Docker Image on DockerHub at *pingidentity/pingdirectory* and visit the *PingIdentity DevOps documentation*.

> ⓘ **Note:**
>
> Only the Directory Proxy Server software is licensed under Ping Identity's end user license agreement, and any other software components contained within the image are licensed solely under the terms of the applicable open source/third-party license.
>
> Ping Identity accepts no responsibility for the performance of any specific virtualization software and in no way guarantees the performance or interoperability of any virtualization software with its products.

Java Runtime Environment
The following Java Development Kit versions are a pre-installation requirement.

- Oracle JDK 8 64-bit
- Oracle JDK 11 LTS 64-bit
- OpenJDK 8 64-bit
- OpenJDK 11
- Corretto 8
- Corretto 11

> ⓘ **Note:**
>
> The *Ping Identity Java Support Policy* applies to your Java Runtime Environment.

Browsers

Administration Console

- Chrome
- Firefox
- Internet Explorer 11 or later

End users

- Chrome
- Edge
- Firefox
- Internet Explorer 11 or later
- Safari

Defining a naming strategy for server locations

The various objects defined in the PingDirectoryProxy Server will be specific to a particular location. Location names are used to define a grouping of PingDirectoryProxy Server products based on physical proximity. For example, a location is most often associated with a single datacenter location. During the installation, assign a location to each server for optimal inter-server behavior. The location assigned to a server within Global Configuration can be referenced by components within the server as well as processes external to the server to satisfy "local" versus "remote" decisions used in replication, load balancing, and failover.

Installing Java

About this task

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

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

Steps

1. Click JDK Download.
2. Go to the *Oracle download site*.
3. On the download page, select the download file for your operating system and accept the license agreement to start the download.

Preparing the operating system

You should make the following changes to your operating system depending on the production environments on which the PingDirectoryProxy Server will run.

Configuring the file descriptor limits

Operating system default file descriptor limitsrestrict the number of PingDirectory Server connections. Change the descriptor limits to allow more connections.

About this task

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

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

After you set the operating system limit, you can configure the number of file descriptors that the server will use either by using a **`NUM_FILE_DESCRIPTORS`** environment variable, or by creating a **`config/num-file-descriptors`** file with a single line such as **`NUM_FILE_DESCRIPTORS=12345`**. If these are not set, the operating system uses the default of 65535 descriptors. This is an optional change you can make if you want to ensure that the server shuts down safely prior to reaching the file descriptor limit.

Steps

1. Display the current `fs.file-max` limit of the system.

   ```
   sysctl fs.file-max
   ```

   The `fs.file-max` limit is the maximum server-wide file limit you can set without tuning the kernel parameters in the `proc` file system.

2. Edit the `/etc/sysctl.conf` file.

   If there is a line that sets the value of the `fs.file-max` property, make sure that its value is set to at least 1.5 times the per-process limit.

   If there is no line that sets a value for this property, add the following to the end of the file:

   ```
   fs.file-max = 100000
   ```

   > ⓘ **Note:**
   >
   > 100000 is just an example here. Specify a value of at least 1.5 times the per-process limit.

3. Display the current hard limit of the system.

   ```
   ulimit -aH
   ```

   The `open files (-n)` value is the maximum number of open files per process limit.

   The value should be set to at least 65535.

4. Edit the `/etc/security/limits.conf` file.

   If the file has lines that set the soft and hard limits for the number of file descriptors, make sure the values are set to 65535. If the lines are not present, add the following lines before `#End of file`, making certain to insert a tab between the columns.

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

   > ⓘ **Note:**
   >
   > The number of open file descriptors is limited by the physical memory available to the host. You can determine this limit with the following command.
   >
   > ```
   > cat /proc/sys/fs/file-max
   > ```
   >
   > If the file-max value is significantly higher than the 65535 limit, consider increasing the file descriptor limit to between 10% and 15% of the system-wide file descriptor limit. For example, if the file-max value is 810752, you could set the file descriptor limit to 100000. If the file-max value is lower than 65535, the host is likely not sized appropriately.

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

   ```
   # ulimit -n
   ```

Results

> ⓘ **Note:**
>
> For RedHat 7 or later, modify the `20-nproc.conf` file to set both the open files and max user processes limits.
>
> ```
> /etc/security/limits.d/20-nproc.conf
>
>  Add or edit the following lines if they do not already exist:
>
>  *          soft    nproc     65536
> ```

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

Enabling the server to listen on privileged ports (Linux)
For your convenience, enable the server to listen on privileged ports while running as a non-root user.

About this task

Linux systems have a mechanism called capabilities that is used to grant specific commands the ability to do things that are normally only allowed for a root account:

- The **`setcap`** command assigns capabilities to an application.
- The **`cap_net_bind_service`** capability enables a service to bind a socket to privileged ports (port numbers less than 1024).

Steps

1. If Java is installed in **`/ds/java`** (and the Java command to run the server is **`/ds/java/bin/java`**), you can grant the **`cap_net_bind_service`** capability to the Java binary with the following command.

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

2. Create the file **`/etc/ld.so.conf.d/libjli.conf`** with the path to the directory that contains the **`libjli.so`** file.

   The java binary needs an additional shared library (**`libjli.so`**) as part of the Java installation. Because this process imposes stricter limits on where the operating system looks for shared libraries to load for commands that have capabilities assigned, it is also necessary to tell the operating system where to look for this library.

   For example, if the Java installation is in **`/ds/java`**, the contents of that file should be:

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

3. To apply the changes, run the following command.

   ```
   $ sudo ldconfig -v
   ```

Setting the file system flushes
Improve Directory Server performance by reducing the span between file system flushes.

About this task

By default, Linux servers running the ext3 file system flush the data to disk every five seconds.

Steps

1. If the Directory Proxy Server is running on a Linux server using the ext3 file system, consider editing the mount options for that file system to include the following command.

   ```
   commit=1
   ```

   This variable changes the flush frequency from five seconds to one second.
2. You should also set the flush frequency in the `/etc/fstab` file.

   Changing the mount command alone does not survive across reboots.

Disabling file system swapping

Because file system swapping can interfere with PingDirectory, disable performance tuning tools like **`tuned`**.

About this task

Steps

1. Sign on as the root user.
2. Add the line `vm.swappiness = 0` to the file `/etc/sysctl.conf`.
3. Restart the system to apply the change.
4. Optional: If you need to tune performance after disabling file system swapping, do the following:
   a. Clone the existing performance profile.
   b. Run **`tuned`**.
   c. Add the line `vm.swappiness = 0` to the file `/usr/lib/tuned/profile-name/tuned.conf`.
   d. To select the updated profile, run **`tuned-adm profile customized_profile`**.
   e. Restart the system to apply the changes.

About editing OS-level environment variables

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

For this reason, the Directory Proxy Server explicitly overrides the values of key environment variables like *<PATH>*, *<LD_LIBRARY_PATH>*, and *<LD_PRELOAD>* to ensure that environment settings used to start the server do not inadvertently impact its behavior.

If you need to edit any of these environment variables, set the values of those variables by manually editing the *<set_environment_vars>* function of the `lib/_script-util.sh` script.

You must stop (bin/stop-server) and re-start (bin/start-server) the server for the change to take effect.

Installing sysstat and pstack (Red Hat)

If you plan to run PingDirectory on a Red Hat Linux system, install a couple of packages, **`sysstat`** and **`pstack`**, that are disabled by default, but are useful for troubleshooting.

About this task

The troubleshooting tool **`collect-support-data`** uses the **`iostat`**, **`mpstat`**, and **`pstack`** utilities to collect monitoring, performance, and stack trace information on the server's processes.

Steps

- To install **`systat`** on your Red Hat system, run the following command.

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

  .

Installing dstat (SUSE Linux)

dstat is a system monitoring tool that can help you troubleshoot PingDirectory servers on SUSE distributions.

About this task

The **collect-support-data** tool uses the **dstat** utility for system monitoring. **dstat** can be obtained from the OpenSUSE project website. The following process shows how to install the **dstat** utility on SUSE Enterprise Linux 11 SP2:

Steps

1. Sign on as the root user.
2. Add the appropriate repository using the **zypper** tool.
3. Install the **dstat** utility.

```
$ zypper install dstat
```

Omitting vm.overcommit_memory

An improperly configured value for the `vm.overcommit_memory` property in the `/etc/sysctl.conf` file might cause the **setup** or **start-server** tool to fail.

About this task

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

Steps

Omit the `vm.overcommit_memory` property in the `/etc/sysctl.conf` file to ensure that enough memory is available for these tools.

Managing system entropy

Linux uses entropy to calculate random data that is used by the system in cryptographic operations.

About this task

Some environments with low entropy might have intermittent performance issues with SSL-based communication. This is more typical on virtual machines but can occur in physical instances as well.

Steps

- Monitor the **kernel.random.entropy_avail** in **sysctl** value for best results.

Setting file system event monitoring (inotify)

An event monitoring tool such as **inotify** can be configured for notifying processes about file system events, including file creation, deletion, and updates.

About this task

Linux limits the number of **inotify** watches a user can receive.

Steps

- To increase the limit, edit **`etc/sysctl.conf`** to add the following line.

```
fs.inotify.max_user_watches = 524288
```

- Run the following command.

```
$ sudo sysctl -w fs.inotify.max_user_watches=524288
```

Tuning the I/O scheduler
Using the correct I/O scheduler can increase performance and reduce the possibility of database timeouts when the system is under extreme write load.

About this task

For file systems running on an SSD or in a virtualized environment, use the recommended **`noop`** scheduler. For all other systems, use the **`deadline`** scheduler.

The procedure for configuring a scheduler to use at startup depends on the version of Linux. See the Linux documentation for your specific version for the correct way to configure this setting.

Steps

1. To determine which scheduler is configured on your system, run the following command.

```
$ cat /sys/block/<block-device>/queue/scheduler
```

2. To change the scheduler on a running system, run the following command.

```
$ echo 'deadline' > /sys/block/sda/queue/scheduler
```

3. To apply the change, restart the system.

Getting the installation packages

Obtain the latest `.zip` release bundle from Ping Identity and extract it to a folder of your choice to begin the installation process.

About this task

The release bundle contains the Directory Proxy Server code, tools, and package documentation.

Complete the following steps to unpack the build distribution.

Steps

1. Download the latest `.zip` distribution of the Directory Proxy Server software.
2. Extract the compressed `.zip` archive file to a directory of your choice.

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

You can set up the Directory Proxy Server.

Sign on to the Administrative Console

After the server is installed, access the Administrative Console, `https://<host>/console/login`, to verify the configuration and manage the server. The root user DN or the common name of a root user DN is required to log into the Administrative Console. For example, if the DN created when the server was

installed is `cn=Directory Manager`, `directory manager`can be used to log into the Administrative Console.

If the Administrative Console needs to run in an external container, such as Tomcat, a separate package can be installed according to that container's documentation. Contact Ping Identity Customer Support for the package location and instructions.

ⓘ **Note:** The session timeout for the console is 24 hours by default. When this duration is exceeded, all inactive users are logged off automatically.

To set a different timeout value, configure the `server.sessionTimeout` application parameter, which specifies the timeout duration in seconds. You can set the value as an init parameter either in the console or on the command line, as shown below.

- Console

  Select **Web Application Extensions**# **Console** and then use the **Init Parameter** field.
- Command line

  The following example uses a value of 1800 seconds (30 minutes).

```
dsconfig set-web-application-extension-prop --no-prompt \
--extension-name Console \
--add init-parameter:server.sessionTimeout=1800
```

For the change to take effect, restart the HTTP(S) Connection Handler or the server itself.

Ping license keys

License keys are required to install, update, and renew all Ping products.

How to obtain a license

To obtain a license key, contact your account representative or use the *Ping Identity licensing portal*.

When do you need a license

A license is required for setting up a new single server instance and can be used site-wide for all servers in an environment. Additionally, you must obtain a new license when updating a server to a new major version, such as when upgrading from 7.3 to 8.0. When cloning a server instance with a valid license, you do not need a new license.

ⓘ **Note:**

The update process displays a prompt for a new license.

How to specify a license

- Specify a license at setup

  You have these options:

  - Use the `--licenseKeyFile` *<path-to-license>* option with **`setup`**.
  - Copy the license file to the server root directory and then run the **`setup`** tool. The tool discovers the license file.

- Specify a license after setup

  Use the Administrative Console or **`dsconfig`** (in the Topology section, select License).

  > ⓘ **Note:** Placing the new license file in the server root directory does not work in this case.

How to view the license status

To view the details of a license, including its expiration, you have these options:

- The server's **`status`** tool
- The Administrative Console's **Status** page (On the **Monitors** tab, search for License.)

License expiration

The server provides a notification as the expiration date approaches.

Before a license expires, obtain a new one and install it by using **`dsconfig`** or the Administrative Console.

> ⓘ **Note:**
>
> An expiring license causes alerts and alarms but does not affect the functionality of the product.

Installing the Directory Proxy Server

When you deploy PingDirectoryProxy Server in a topology, you generally deploy them in pairs. These pairs are configured identically except for their host name, port name, and possibly their location.

To help administrators easily install identical proxies, the Directory Proxy Server allows you to clone a proxy configuration. First, you install a Directory Proxy Server using the **`setup`** tool. Then, you configure it using the **`create-initial-proxy-config`** tool described in *Using the create-initial-proxy-config Tool*. Finally, you run the **`setup`** tool on subsequent servers, indicating that you want to clone the configuration on a peer server.

The following sections describe the setup tool in more detail, and tell you how to install first and subsequent proxies in your topology.

About the setup tool

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

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

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

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

Installing the first Directory Proxy Server in interactive mode

The **`setup`** tool provides an interactive text-based interface to install a Directory Proxy Server instance.

Installing the first Directory Proxy Server in interactive mode

Steps

1. Change to the server root directory.

```
cd Directory Proxy Server
```

2. Use the **`setup`** command.

```
$ ./setup
```

3. Read the Ping Identity End-User License Agreement, and type `yes` to continue.
4. Press **Enter** to accept the default of `no` in response to adding this new server to an existing topology.

```
Would you like to add this server to an existing Directory Proxy Server
 topology? (yes / no) [no]:
```

5. Enter the fully qualified host name for this server, or press **Enter** to accept the default.
6. Create the initial root user DN for this server, or press **Enter** to accept the default.
7. Enter and confirm a password for this account.
8. To enable the Directory Proxy Server services (Configuration, Documentation, and Directory REST API) and Administrative Console over HTTPS, press **Enter** to accept the default. After setup, individual services can be enabled or disabled by configuring the HTTPS Connection Handler.
9. Enter the port on which the Directory Proxy Server should accept connections from HTTPS clients, press **Enter** to accept the default.
10. Enter the port on which the Directory Proxy Server should accept connections from LDAP clients, press **Enter** to accept the default.
11. The next two options enable LDAPS and StartTLS. Press **Enter** to accept the default (yes), or type **no**. If either are enabled, certificate options are required. To use the Java Keystore or the PKCS#12 keystore, the keystore path and the key PIN are required. To use the PKCS#11 token, only the key PIN is required.
12. Choose a certificate server option:

```
Certificate server options:
  1) Generate self-signed certificate (recommended for testing purposes
 only)
  2) Use an existing certificate located on a Java Keystore (JKS)
  3) Use an existing certificate located on a PKCS#12 keystore
  4) Use an existing certificate on a PKCS#11 token
```

13. Choose the desired encryption for backups and log files from the choices provided:
    - Encrypt data with a key generated from an interactively provided passphrase. Using a passphrase (obtained interactively or read from a file) is the recommended approach for new deployments, and you should use the same encryption passphrase when setting up each server in the topology.
    - Encrypt data with a key generated from a passphrase read from a file.
    - Encrypt data with a randomly generated key. This option is primarily intended for testing purposes, especially when only testing with a single instance, or if you intend to import the resulting encryption settings definition into other instances in the topology.
    - Encrypt data with an imported encryption settings definition. This option is recommended if you are adding a new instance to an existing topology that has older server instances with data encryption enabled.
    - Do not encrypt server data.
14. To configure your Directory Proxy Server to use entry balancing, type `yes`, or accept the default `no`. In an entry balancing environment, entries immediately beneath the balancing base DN are divided into disjoint subsets. Each subset of data is handled by a separate set of one or more directory server

instances, which replicate this subset of data between themselves. Choosing `yes` will enable more memory be allocated to the server and tools.

15. Choose the option for the amount of memory to assign to this server.
16. Enter an option to set up the server with the current configuration, provide new parameters, or cancel.
17. After setup is complete, choose the next configuration option.

```
This server is now ready for configuration What would you like to do?

   1) Start 'create-initial-proxy-config' to create a basic
      initial configuration (recommended for new users)
   2) Start 'dsconfig' to create a configuration from scratch
   3) Quit

Enter choice [1]:
```

Installing additional Directory Proxy Server instances in interactive mode

About this task

The **`setup`** tool provides an interactive text-based interface to install a Directory Proxy Server instance that clones a previously installed Directory Proxy Server instance.

Steps

1. Change to the server root directory.

```
cd Directory Proxy Server
```

2. Use the **`setup`** command.

```
$ ./setup
```

3. Read the Ping Identity End-User License Agreement, and type `yes` to continue.
4. Enter `yes` in response to add this new server to an existing topology.

```
Would you like to add this server to an existing Directory Proxy Server
 topology? (yes / no) [no]: yes
```

5. Enter the host name of the Directory Proxy Server from which configuration settings are copied during setup.

```
Enter the host name of the peer Directory Proxy Server from which you would
 like
to copy configuration settings. [proxy.example.com]:
```

6. Type the port number of the peer Directory Proxy Server from which configuration settings are copied during setup. You can press **Enter** to accept the default port, which is 389.

```
Enter the port of the peer Directory Proxy Server [389]:
```

7. Enter the option corresponding to the type of connection you want to use to connect to the peer Directory Proxy Server.

```
How would you like to connect to the peer Directory Proxy Server?
  1) None
  2) SSL
  3) StartTLS

Enter choice [1]:
```

8. Type the root user DN of the peer Directory Proxy Server, or press **Enter** to accept the default (cn=Directory Manager), and then type and confirm the root user password.

```
Enter the manager account DN for the peer Directory Proxy Server
 [cn=Directory Manager]:
Enter the password for cn=Directory Manager:
```

9. Enter the host name of the new local Directory Proxy Server.

```
Enter the fully qualified host name or IP address of the local host
 [proxy.example.com]:
```

10. Choose the location of your new Directory Proxy Server instance or enter a new one.
11. Enter an option to set up the server with the current configuration, provide new parameters, or cancel.
12. After setup is complete, choose the next configuration option.

Installing the first Directory Proxy Server in non-interactive mode

About this task

You can run the **`setup`** command in non-interactive mode to automate the installation process using a script or to run the command directly from the command line. If there is a missing or incorrect argument, the **`setup`** tool fails and aborts the process.

The **`setup`** tool automatically chooses the maximum heap size. You can manually tune the maximum amount of memory devoted to the server's process heap using the `--maxHeapSize` option. The `--maxHeapSize` argument is only valid if the `--entryBalancing` or `--aggressiveJVMTuning` options are also present.

For example, use the `--aggressiveJVMTuning` option to set the maximum amount of memory used by the Directory Proxy Server and tools as follows:

```
--aggressiveJVMTuning --maxHeapSize 256m
```

If you are using entry balancing, tune the amount of memory devoted to the Directory Proxy Server using the `--entryBalancing` option as follows:

```
--entryBalancing --maxHeapSize 1g
```

The amount of memory allowed when using the `--entryBalancing` option is calculated and depends on the amount of system memory available. If you are using entry balancing and also want the tools to get more memory, include both the `--entryBalancing` and the `--aggressiveJVMTuning` options.

```
--entryBalancing --aggressiveJVMTuning --maxHeapSize 1g
```

If you have already configured a truststore, you can also use the setup tool to enable security. The following example enables security, both SSL and StartTLS. It also specifies a JKS Keystore and Truststore that define the server certificate and trusted CA. The passwords for the keystore files are defined in the corresponding .pin files, where the password is written on the first line of the file. The values in the .pin files will be copied to the `server-root/config` directory in the `keystore.pin` file.

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

```
$ env JAVA_HOME=/ds/java ./setup --cli \
   --no-prompt --rootUserDN "cn=Directory Manager" \
   --rootUserPassword "password" --ldapPort 389 \
   --enableStartTLS --ldapsPort 636 \
   --useJavaKeystore /path/to/devkeystore.jks \
   --keyStorePasswordFile /path/to/devkeystore.pin \
   --certNickName server-cert \
```

```
 --useJavaTrustStore /path/to/devtruststore.jks \
 --trustStorePasswordFile /path/to/devtruststore.pin \
 --acceptLicense
```

Steps

- Use **setup** with the `--no-prompt` option. The command uses the default root user DN (`cn=Director Manager`) with the specified `--rootUserPassword` option. You must include the `--acceptLicense` option or the **setup** tool will generate an error message.

```
$ env JAVA_HOME=/ds/java ./setup --no-prompt \
--rootUserDN "cn=Directory Manager" \
--rootUserPassword "password" --ldapPort 389 \
--acceptLicense
```

Installing additional Directory Proxy Server in non-interactive mode

About this task

You can run the **setup** command in non-interactive mode to automate the installation process using a script or to run the command directly from the command line. If there is a missing or incorrect argument, the **setup** tool fails and aborts the process.

Steps

- Use **setup** with the `--no-prompt` option.

```
$ env JAVA_HOME=/ds/java ./setup --cli --no-prompt \
--rootUserDN "cn=Directory Manager" \
--rootUserPassword "password" --ldapPort 1389 \
--localHostName proxy2.example.com \
--peerHostName proxy1.example.com --peerPort 389 \
--peerUseNoSecurity --acceptLicense --location austin1
```

Installing the Directory Proxy Server with a truststore in non-interactive mode

About this task

If you have already configured a trust store, you can also use the **setup** tool to enable security. The following example enables SSL security. It also specifies a JKS Keystore and truststore that define the server certificate and trusted CA. The passwords for the keystore files are defined in the corresponding .pin files, where the password is written on the first line of the file. The values in the .pin files will be copied to the `server-root/config` directory in the `keystore.pin` and `truststore.pin` files.

ⓘ **Note:** The password to the private key within the keystore is expected to be the same as the password to the keystore. If this is not the case, the private key password can be defined within the or **dsconfig** by editing the Key Manager Provider standard configuration object.

Steps

- Run the **setup** tool to install with a truststore.

```
$ env JAVA_HOME=/ds/java ./setup --cli \
  --no-prompt --rootUserDN "cn=Directory Manager" \
  --rootUserPassword "password" \
  --ldapPort 389 --ldapsPort 636 \
  --useJavaKeystore /path/to/devkeystore.jks \
  --keyStorePasswordFile /path/to/devkeystore.pin \
  --certNickName server-cert \
```

```
  --useJavaTrustStore /path/to/devtruststore.jks \
  --acceptLicense

In order to update the trust store, the password must be provided

See 'prepare-external-server --help' for general overview

Testing connection to ds-east-01.example.com:1636 ..... Done
Testing 'cn=Proxy User,cn=Root DNs,cn=config' access .....
Created 'cn=Proxy User,cn=Root DNs,cn=config'

Testing 'cn=Proxy User,cn=Root DNs,cn=config' access ..... Done
Testing 'cn=Proxy User,cn=Root DNs,cn=config' privileges ..... Done
Verifying backend 'dc=example,dc=com' ..... Done
```

Directory Proxy Server folder layout

Once you have unzipped the Directory Proxy Server distribution file, the following folders and command-line utilities are available.

Layout of the Directory Proxy Server Folders

Directories/Files/Tools	Description
License.txt	Licensing agreement for the Directory Proxy Server.
README	README file that describes the steps to set up and start the Directory Proxy Server.
bak	Stores the physical backup files used with the **`backup`** command-line tool.
bat	Stores Windows-based command-line tools for the Directory Proxy Server.
bin	Stores UNIX/Linux-based command-line tools for the Directory Proxy Server.
classes	Stores any external classes for server extensions.
collector	Used by the server to make monitored statistics available to the Data Metrics Server.
config	Stores the configuration files for the backends (admin, config) as well as the directories for messages, schema, tools, and updates.
docs	Provides the product documentation.
import-tmp	Stores temporary imported items.
ldif	Stores any LDIF files that you may have created or imported.
legal-notices	Stores any legal notices for dependent software used with the Directory Proxy Server.
lib	Stores any scripts, jar, and library files needed for the server and its extensions.
locks	Stores any lock files in the backends.
logs	Stores log files for the Directory Proxy Server.
metrics	Stores the metrics that can be gathered for this server and surfaced in the Data Metrics Server.
resource	Stores the MIB files for SNMP and can include ldif files, make-ldif templates, schema files, dsconfig batch files, and other items for configuring or managing the server.
revert-update	The **`revert-update`** tool for UNIX/Linux systems.

Directories/Files/Tools	Description
revert-update.bat	The **revert-update** tool for Windows systems.
setup	The **setup** tool for UNIX/Linux systems.
setup.bat	The **setup** tool for Windows systems.
scim-data-tmp	Used to create temporary files containing SCIM request data.
uninstall	The **uninstall** tool for UNIX/Linux systems.
uninstall.bat	The **uninstall** tool for Windows systems.
update	The **update** tool for UNIX/Linux systems.
update.bat	The **update** tool for Windows systems.
Velocity	Stores any customized Velocity templates and other artifacts (CSS, Javascript, images), or Velocity applications hosted by the server.

Uninstalling the Server

The Directory Proxy Server provides an **uninstall** command-line utility for quick removal of the code base.

To uninstall a server instance, run the **setup** tool in interactive command-line, or non-interactive command-line mode.

Interactive command-line mode

Interactive command-line mode is a text-based interface that prompts the user for input. To run this mode, use the **bin/uninstall** command with the `--cli` option.

Non-interactive command-line mode

Non-interactive command-line mode suppresses progress information from being written to standard output during processing, except for fatal errors. To run this mode, use the **bin/uninstall** command with the `--no-prompt` option.

ⓘ **Note:**

For standalone installations with a single Directory Proxy Server instance, you can also manually remove the Directory Proxy Server by stopping the server and recursively deleting the directory and subdirectories, as in the following example.

```
$ rm -rf /ds/PingDirectoryProxy
```

Uninstalling the server in interactive mode

Interactive mode uses a text-based, command-line interface to help you remove your instance.

About this task
If **uninstall** cannot remove all of the Directory Proxy Server files, the **uninstall** tool generates a message with a list of the files and directories that must be manually deleted. The **uninstall** command must be run as either the root user or the same user (or role) that installed the Directory Proxy Server.

Steps

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

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

2. Select the components to be removed.
   - Remove all components - If you want to remove all components, press **Enter** to accept the default (remove all).
   - Select the components to be removed - If you do not want to remove all components, enter the option to specify the removal of only specific components.

     For each type of server component, press **Enter** to remove it or enter `no` to keep it.
3. If the Directory Proxy Server is part of a replication topology, enter `yes` to provide your authentication credentials (Global Administrator ID and password). If you are uninstalling a standalone server, continue to step 7.
4. Enter the Global Administrator ID and password to remove the references to this server in other replicated servers, and then enter or verify the host name or IP address for the server that you are uninstalling.
5. Select how you want to trust the server certificate if you have set up SSL or StartTLS. Press **Enter** to accept the default.

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

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

Enter choice [3]:
```

6. View the logs for any remaining files and manually remove any remaining files or directories.

   If your Directory Proxy Server is running, the server shuts down before continuing the uninstall process. The uninstall action processes the removal requests before completing.

Uninstalling the server in non-interactive mode

The **uninstall** utility provides a non-interactive method to enter the command with the `--no-prompt` option.

About this task
Another useful argument is the `--forceOnError` option that continues the uninstall process when an error is encountered. If an option is incorrectly entered or if a required option is omitted and the `--forceOnError` option is not used, the command fails and aborts.

Steps

1. From the server root directory, run **uninstall** tool with the `--remove-all` option to remove all of the Directory Proxy Server's libraries.

   The optional `--quiet` option suppresses output information.

   The following command assumes that the Directory Proxy Server is standalone and not part of a replication topology.

```
$ ./uninstall --cli --remove-all --no-prompt --quiet --forceOnError
```

2. If any files or directories remain, manually remove them.

Uninstalling selected components in non-interactive mode

Steps

1. From the server root directory, run **`uninstall`** with the `--backup-files` option to remove the Directory Proxy Server's backup files.

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

2. To view the other options available to remove specific components, use the `--help` or `-H` option.

Upgrading the Directory Proxy Server

Ping Identity issues new software builds periodically and distributes the software package in zip format. Administrators can use the Directory Proxy Server's **`update`** utility to update the current server code with the latest features and bug fixes. To update the Directory Proxy Server to a newer version, download the build package, and then unzip the new server package on the same host as the server that you wish to update. Before upgrading a server, you should ensure that it is capable of starting without severe or fatal errors.

> ⓘ **Tip:**
>
> For additional considerations, see *Planning your upgrade*.

During an update process, the updater checks a manifest file that contains a MD5 checksum of each file in its original state when installed from zip. Next, it compares the checksum of the new server files to that of the old server. Any files that have different checksums will be updated. For files that predates the manifest file generation, the file is backed up and replaced. The updater also logs all file changes in the history directory to tell what files have been changed.

For schema updates, the **`update`** tool preserves any custom schema definitions (`99-user.ldif`). For any default schema element changes, if any, the updater will warn the user about this condition and then create a patch schema file and copy it into the server's schema directory. For configuration files, the update tool preserves the configuration file, `config.ldif`, unless new configuration options must be added to the Directory Proxy Server.

Once the updater finishes its processing, it checks if the newly updated server starts without any fatal errors. If an error occurs during the update process, the **`update`** tool reverts the server root instance to the server state prior to the update.

Upgrade overview and considerations

Considerations when upgrading to 8.2.0.0

This upgrade moves to Jetty 9.4. As a result, the HTTPS connection handler will no longer support TLS_RSA ciphers by default. If you use any legacy HTTPS clients that still require TLS_RSA ciphers, modify the `ssl-cipher-suite` property of the HTTPS Connection Handler to include them.

Upgrading servers in a topology

An update to the current PingDirectory Server release includes the introduction of a topology registry, which stores information that was stored previously in the admin backend, such as server instances, instance and secret keys, server groups, and administrator user accounts.

Before you begin

Before you can update and migrate the admin backend, the following conditions must be satisfied:

- Run the update tool and provide LDAP authentication options to the peer servers of the server being updated. The **`update`** tool connects to the peer servers of the server being updated to obtain the necessary information to populate the topology registry.
- On every server in the topology, configure the encryption protocol requested, either plain, TLS, StartTLS, or SASL, for an LDAP connection.
- Ensure the LDAP credentials are present on every server in the topology.
- Ensure the users related to the LDAP credentials have permissions to read from the admin backend and the config backend of every server in the topology. For example, you can use a root DN user that has `inherit-default-privileges` set to true, such as the `cn=Directory Manager` user, that exists on every server.
- The instance name is set on every server and is unique across all servers in the topology. The instance name is a server's identifier in the topology. After all servers in the topology have been updated, each server is uniquely identified by its instance name. After the name is set, it cannot be changed.

> ⓘ **Tip:**
>
> If needed, use the following command to set the instance name of a server prior to the update.
>
> ```
> $ bin/dsconfig set-global-configuration-prop \
>   --set instance-name:uniqueName
> ```

> ⓘ **Important:**
>
> This information is only applicable when updating from a server version that uses the admin backend to a server version that uses the topology registry. This is only the case when updating from a server version earlier than a 7.x to a 7.x version and is not applicable to any updates to a server version of 8.x or later.

About this task
Upgrade to a server that uses a topology registry to store network configuration, administrator user accounts, and keys information, as well as migrate the information from the previous server version.

Steps

1. To make clustered configuration changes in a mixed-version cluster, choose one of the following options:
   - Update each server to the same version.
   - Temporarily split up the cluster by changing the `cluster-name` property on the server instance configuration objects.

   > ⓘ **Note:**
   >
   > Changes to clustered configurations are not allowed in mixed-version clusters. This applies to configuration in the `cn=Cluster,cn=config` subtree and only applies to servers with matching cluster names.
2. Make clustered configuration changes again to mirror these changes across the topology.
3. Run the **`dsframework`** command on the server being updated.

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

```
  --set startTLSEnabled:true
```

> **Important:**
>
> To run this command, you must have enabled or fixed the configuration of the LDAP Connection Handlers to support the desired connection security protocol on each server, so that its admin backend has the most up-to-date information.

Before allowing the update, the **update** tool verifies that the following conditions are satisfied on every server in the topology:

- When the first server is being updated, all other servers in the topology are online.
- When updating additional servers, all topology information was obtained from one of the servers that has already been updated.
- The users related to the provided LDAP credentials have read permissions to the config and admin backends of the peer servers.
- The instance name is set on every server and is unique across all servers in the topology.

> **Note:**
>
> If any of these conditions or those listed in the preceding Before you begin section are not satisfied, the **update** tool lists all of the errors encountered for each server and provides instructions on how to fix them.

When all of these conditions are met, the cluster-wide configuration is synchronized on all servers in the topology.

> **Note:**
>
> Older versions have some topology configuration under the **cn=cluster,cn=config** JSON attribute and field constraints. These items do not support mirrored cluster-wide configuration data. In an update, avoid custom configuration changes on a server being overwritten with the configuration on the mirrored subtree master. If additional configuration steps are needed, see step 4.

4. To synchronize the cluster-wide configuration data across all servers in the topology, run the **config-diff** tool on each pair of servers to determine the differences, and use **dsconfig** to update each instance using the **config-diff** output.

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

Upgrading the Directory Proxy Server

About this task

Assume that an existing version of the Directory Proxy Server is stored at PingDirectoryProxy-old, which you want to update.

Steps

1. Make sure you have complete, readable backup of the existing system before upgrading the Directory Proxy Server build. Also, make sure you have a clear backout plan and schedule.
2. Download the latest version of the PingDirectoryProxy Server software and unzip the file. For this example, let's assume the new server is located in the PingDirectoryProxy-new directory.
3. Check the version number of the newly downloaded Directory Proxy Server instance using the `--version` option on any command-line utility. For example, you should see the latest revision number.

```
$ PingDirectoryProxy-new/setup --version PingDirectoryProxy Server
                  8.0.0.0
Build 20110432006092 Revision 9235
```

4. Use the **`update`** tool of the newly unzipped build to update the Directory Proxy Server code. Make sure to specify the Directory Proxy Server instance that you are upgrading with the `--serverRoot` option. The Directory Proxy Server must be stopped for this update to be applied.

```
$ PingDirectoryProxy-new/update --serverRoot PingDirectoryProxy-old
```

ⓘ **Note:** The PingDirectoryProxy Server provides a web console called the Administrative Console, to configure and monitor the server. If you update the Directory Proxy Server version, you should also update the Administrative Console.

5. View the log file to see which files were changed. The log file is located in the `<server-root>/history` directory. For example, the file will be labelled with the Directory Proxy Server version number and revision.

```
$ view <server-root>/history/1272307020420-8.0.0.0.9235/update.log
```

Reverting an update

After PingDirectoryProxy Server has been updated, you can revert to the last version or one level back using the **`revert-update`** tool.

About this task

The **`revert-update`** tool accesses a log of file actions taken by the updater to put the file system back to its prior state. If you have run multiple updates, you can run the **`revert-update`** tool multiple times to revert to each prior update sequentially. You can only revert back one level. For example, if you have run the update twice since first installing PingDirectoryProxy Server, you can run the **`revert-update`** command to revert to its previous state, then run the **`revert-update`** command again to return to its original state. The following steps detail reverting from 7.x to a version prior to 7.0.

When starting up the server for the first time after a revert has been run, and the necessary extra steps have been completed, the server displays warnings about "offline configuration changes," but they are not critical and do not appear on subsequent start ups.

To revert to the most recent server version:

Steps

- Use **`revert-update`** in the server root directory to revert back to the most recent version of the server.

```
$ PingDirectoryProxy-old/revert-update
```

Next steps

Reverting from the 7.0 version or later to a version earlier than 7.0 using the **revert-update** command might require extra steps. This is also the case when updating or reverting from a version earlier than 6.2.0.2 to a 6.2.0.2 version or later. These steps are listed when the **update** and **revert-update** tool are run as well. Depending on your installation and configuration, you might need to perform one or more of the following tasks:

- When updating or reverting from 6.2.0.2 or later to a version earlier than 6.2.0.2, indexes might need to be rebuilt. Older versions of the server use an incompatible format for Local DB Composite Indexes. To update a server with composite indexes in the previous format, delete these indexes and re-run the update. After the update is complete, recreate the indexes and rebuild the indexes using the **rebuild-index** tool. The command for recreating an index is in the `Undo` portion of the `logs/config-audit.log` file. To later revert to an older version, delete and recreate those composite indexes again after the revert process completes.
- When updating to 7.x for the first time, instance names must be set for each server in the topology if they were not previously set. This is done with the following **dsconfig** command.

  ```
  $ bin/dsconfig --bindDN "cn=Directory Manager" \
    --bindPassword secret \
    --no-prompt set-global-configuration-prop \
    --set instance-name:<name>
  ```

- Topology information such as server instances, instance and secret keys, server groups, and administrator users have moved to the topology portion of the configuration from the **admin** backend. As long as new servers are not added to the topology after this update, you can use the **revert-update** command to return to the previous version. However, if new servers are added, then the restored **admin** backend of this server does not contain information about the new servers, and the local server cannot communicate with any other servers in the topology. New servers should not be added to the topology if reverting this update is a possibility.
- If new servers were added to the topology after the update, the new servers must be temporarily removed from the topology until all servers have been reverted to the previous version.
- When a server is reverted to a version earlier than 7.x, any servers in the topology using the topology portion of the configuration rather than the **admin** backend must know that the reverted server was downgraded to the **admin** backend. To do this, run the following **dsconfig** command on one of the servers that has not been reverted.

  ```
  $ bin/dsconfig set-server-instance-prop \
    --instance-name <Reverted server instance name> \
    --set server-version:<Version to which server is reverted>
  ```

- If the topology does not have a master server when this command is run, the **revert-update** will not succeed. In this case, you must make one of the remaining updated servers in the topology the master with the following command. This enables the chosen instance to run the **revert-update** command successfully.

  ```
  $ bin/dsconfig set-global-configuration-prop \
    --set force-as-master-for-mirrored-data:true
  ```

- The 7.x server version includes database changes that are not compatible with previous server versions, such as 6.x or earlier. To later revert to an older version, you must export the data to LDIF before performing the reversion. Then, re-import the data after the revert process completes. In addition, you must delete the `changelogDb/` and `db/changelog/` directories in the reverted server root after the revert process completes.

Getting Started with Directory Proxy Server

Running the server

Run the server as a background or foreground process.

Steps

1. To start the Directory Proxy Server, run the bin/start-server command on UNIX or Linux systems.

   An analogous command is in the `bat` folder on Microsoft Windows systems.

   > ⓘ **Note:**
   >
   > The bin/start-server command starts the Directory Proxy Server as a background process when no options are specified.

2. Optional: To run the Directory Proxy Server as a foreground process, use the bin/start-server command with the `--nodetach` option.

Starting the Directory Proxy Server

Start the directory server using the terminal.

Steps

- To start the server, on the terminal run bin/start-server.

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

Running the server as a foreground process

Use the terminal to run the PingDirectory service as a foreground process.

Steps

1. To launch the Directory Proxy Server as a foreground process, open the terminal and enter bin/start-server with the `--nodetach` option.

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

2. To stop the Directory Proxy Server:
   - Press **CNTRL+C** in the terminal window where the server is running.
   - Run the bin/stop-server command from another terminal window.

Starting the server at boot time

By default,PingDirectoryProxy Server does not start automatically when the system is booted. Instead, you must manually start it with the bin/start-server command.

About this task

To configure the Directory Proxy Server to start automatically when the system boots, use the **create-systemd-script** utility to create a script, or create the script manually.

Steps

1. Create the service unit configuration file in a temporary location, where "ds" is the user running PingDirectoryProxy.

```
$ bin/create-systemd-script \
     --outputFile /tmp/ping-directory.service \
     --userName ds
```

2. As a root user, copy the **ping-directory.service** configuration file into the `/etc/systemd/system` directory.
3. To read the new configuration file, reload **systemd**.

```
$ systemctl daemon-reload
```

4. To start the PingDirectoryProxy, run the **start** command.

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

5. To configure the PingDirectoryProxy to start automatically when the system boots, run the **enable** command.

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

6. Log out as root.

   To perform this task on an RC system, create the startup script with **bin/create-rc-script** and move it to the `/etc/init.d` directory.

> ⓘ **Note:**
>
> Create symlinks to this script from the `/etc/rc3.d` directory (starting with an “S” to ensure that the server is started) and `/etc/rc0.d` directory (starting with a “K” to ensure that the server is stopped).

Signing on to the Administrative Console

After the server is installed, access the administrative console, **https://hostname:HTTPport/console/login**, to verify the configuration and manage the server.

To sign on to the administrative console, use the initial root user DN specified during setup (by default **cn=Directory Manager**).

You can use the **dsconfig** command or the administrative console to create additional root DN users in **cn=Root DNs,cn=config**. These new users require the fully qualified DN as the username, such as **cn=new-admin,cn=Root DNs,cn=config**. To use a simple username (without the **cn=** prefix) for signing on to the administrative console, the root DN user must have the **alternate-bind-dn** attribute configured with an alternate name, such as "admin."

The default link to the administrative console is **https://hostname:HTTPport/console/login**.

If you need to run the administrative console in an external container, such as Tomcat, you can install a separate package (`/server-root/resource/admin-console.zip`) according to that container's documentation.

> ⓘ **Note:**
>
> The default session timeout for the console is 24 hours. When this duration is exceeded, all inactive users are signed off automatically.
>
> To set a different timeout value, configure the `server.sessionTimeout` application parameter, which specifies the timeout duration in seconds. You can set the value as an init parameter either in the console or on the command line, as shown below.

For changes to take effect, restart the HTTP(S) Connection Handler or the server.

Console

1. Go to **Web Application Extensions**# **Console**.
2. Use the **Init Parameter** field.

Command line

- The following example uses a value of 1800 seconds (30 minutes).

```
dsconfig set-web-application-extension-prop --no-prompt \
--extension-name Console \
--add init-parameter:server.sessionTimeout=1800
```

Stopping the Directory Proxy Server

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

About this task

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

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

Steps

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

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

Scheduling a server shutdown

Schedule a server shutdown.

Steps

- To schedule a server shutdown, use the **`bin/stop-server`** tool with the `--stopTime YYYYMMDDhhmmss` option .

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

```
$ bin/stop-server --stopTime 20120606134500Z --port 1389 \
  --bindDN "uid=admin,dc=example,dc=com" --bindPassword secret \
  --stopReason "Scheduled offline maintenance"
```

Restarting the server

Restart the Directory Proxy Server.

About this task

Running the command is equivalent to shutting down the server, exiting the Java Virtual Machine (JVM) session, and then starting up again.

Steps

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

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

Running the server as a Microsoft Windows service

The server can run as a Windows service on Windows Server 2012 R2 and Windows Server 2016. This enables signing-out of a machine without stopping the server.

Registering the server as a Windows service

Register the server as a Windows service using the Windows command prompt.

About this task

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

Steps

1. Stop the server with `bin/stop-server`.

   You cannot register a server while it is running.
2. To register the server as a service, from a Windows command prompt, run `bat/register-windows-service.bat`.
3. After registration, start the server from the Windows Services Control Panel or with the `bat/start-server.bat` command.

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

Running multiple service instances

Only one instance of a particular service can run at one time.

About this task

Services are distinguished by the `wrapper.name` property in the `<server-root>/config/wrapper-product.conf` file.

Steps

1. To run additional service instances, change the `wrapper.name` property on each additional instance.
2. Add or change descriptions of the service in the `wrapper-product.conf` file.

Deregistering and uninstalling services
To uninstall a service, you must first deregister it.

About this task

While a server is registered as a service, it cannot run as a non-service process or be uninstalled.

Steps

1. To remove the service from the Windows registry, use the `bat/deregister-windows-service.bat` file.
2. To uninstall the server, run the `uninstall.bat` script.

Configuring log files for services

About this task

The log files are stored in `<server-root>/logs`, and file names begin with `windows-service-wrapper`. They are configured to rotate each time the wrapper starts or the file reaches its maximum size. Only the last three log files are retained.

Steps

These configurations can be changed in the `<server-root>/config/wrapper.conf` file.

Configuring the Directory Proxy Server

Once you have initially configured the PingDirectoryProxy Server, you can manage your deployment using the configuration framework and management tools. This chapter briefly describes these tools and provides procedures to help you maintain and update your deployment.

About the configuration tools

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

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

Using the create-initial-proxy-config tool

The **`create-initial-proxy-config`** tool can be used to initially configure the Directory Proxy Server. We strongly recommend that you use the **`create-initial-proxy-config`** tool for your initial Directory Proxy Server configuration. This tool prompts you for basic information about your topology, including external servers, their locations, and credentials for communicating with them. Once the configuration is complete, the tool writes the configuration to a **`dsconfig`** batch file and allows you to apply the configuration to the local Directory Proxy Server. The tool assumes the following about your topology:

- All servers are accessible through a single user account. This user account must be a root user that is not generally accessible to clients to avoid inadvertent changes, deletions, or backend server availability issues due to reimporting data.

- All servers support the same type of communication security.
- All external servers are any combination of Ping Identity Directory Server, Sun Directory Server, or Red Hat (including Fedora and 389) instances.

If your topology does have these characteristics, you can use the tool to define a basic configuration that is saved to a **dsconfig** batch file. You can then run the **dsconfig** tool to fine-tune the configuration. You can also use this tool to configure an entry balancing configuration, which allows you to automatically spread entries below a common parent among multiple sets of directory servers for improved scalability and performance.

The **create-initial-proxy-config** tool produces a log file called `create-initial-proxy-config.log` that is stored in the local Directory Proxy Server's `logs` directory.

You can only run the **create-initial-proxy-config** tool once for the initial configuration of each Directory Proxy Server instance. To tune your configuration, use the **dsconfig** tool. When installing a second Directory Proxy Server, it will not be necessary to run the **create-initial-proxy-config** tool again, as the Directory Proxy Server setup has the ability to clone the settings from an existing Directory Proxy Server.

This section describes how to use this tool to configure a standard Directory Proxy Server deployment as well as an entry balancing configuration.

Configuring a standard Directory Proxy Server deployment

About this task

This section describes how to install a standard Directory Proxy Server deployment using the **create-initial-proxy-config** tool. Remember that you deploy the Directory Proxy Server in pairs. Each pair should be configured identically except for their host name, port, and possibly their location.

Steps

1. After initial installation, select the number to start the **create-initial-proxy-config** tool automatically. Otherwise, run it manually at the command line from the server root directory, <server-root>/PingDirectoryProxy.

```
$ ./bin/create-initial-proxy-config
```

2. The initial proxy configuration presents the assumptions about the underlying Directory Server backend servers. If the servers do not meet the requirements, then you can enter "no" to quit the process.

```
Some assumptions are made about the topology in order to keep this tool
 simple:

 1) all servers will be accessible via a single user account
 2) all servers support the same communication security type
 3) all servers are PingDirectoryProxy Server, Directory Server,
    Java System 5.x, 6.x, or 7.x, or Red Hat (including Fedora and 389)
    directory servers

If your topology does not have these characteristics you can use this tool
 to define a basic configuration and then use the 'dsconfig' tool or the
 Administrative Console to
fine tune the configuration.

Continue? (yes / no) [yes]:
```

3. Enter the DN for the Directory Proxy Server user account, then enter and confirm the password for this account. Note that you should not use `cn=Directory Manager` account for communication between the Directory Proxy Server and the Directory Server. For security reasons, the account used to communicate between the Directory Proxy Server and the Directory Server should not be directly

accessible by clients accessing the Directory Proxy Server. For more information about this account, see *Configuring LDAP External Servers*.

```
Enter the DN of the proxy user account [cn=Proxy User,cn=Root
 DNs,cn=config]:
Enter the password for 'cn=Proxy User,cn=Root DNs,cn=config':
Confirm the password for 'cn=Proxy User,cn=Root DNs,cn=config':
```

4. Specify whether you will be using secure communication with the Directory Server instances.

```
>>>> External Server Communication Security

Specify the type of security that the Directory Proxy Server will use when
 communicating with directory server instances:

   1)  None
   2)  SSL
   3)  StartTLS

   b)  back
   q)  quit

Enter choice [1]:
```

5. Specify the base DNs of the Directory Server instances that will be accessed through the Directory Proxy Server. The Directory Proxy Server will create subtree views using each base DN to define portions of the external servers' DIT available for client access. You can specify more than one base DN. Press **Enter** when you have finished specifying the DN(s).

```
Enter a base DN of the directory server instances that will be accessed
 through the Identity Proxy:

    b)  back
    q)  quit

Enter a DN or choose a menu item [dc=example,dc=com]:
```

6. Next, specify if the entries under your defined subtree view will be split across multiple servers in an entry balanced deployment. For this example, press Enter to accept the default ("no").
7. Define a location for your server, such as the name of your data center or the city where the server is located. This example illustrates defining a location named `east`.

```
Enter a location name or choose a menu item: east
```

8. If you defined more than one location, specify the location that contains the Directory Proxy Server itself.

```
Choose the location for this Directory Proxy Server

    1)  east
    2)  west

    b)  back
    q)  quit

Enter choice [1]: 1
```

9. Define the hostname:port used by the LDAP external servers. If you have specified more than one location, you will go through this process for each location.

```
Enter a host:port or choose a menu item [localhost:389]: ldap-
east-01.example.com:389
```

10. After each step, the server will attempt to prepare each external server by testing the communication between the Directory Proxy Server and the Directory Server. Select the option "Yes, and all

subsequent servers" to indicate that you want the tool to create a proxy user account on all of your LDAP external servers within that location.

```
Would you like to prepare ldap-east-01.example.com:389 for access by the
 Directory Proxy Server?

    1) Yes
    2) No
    3) Yes, and all subsequent servers
    4) No, and all subsequent servers

Enter choice [1]: 3
```

11. If the proxy user account did not previously exist on your LDAP external server, create the account by connecting as `cn=Directory Manager`.

```
Would you like to create or modify root user 'cn=Proxy User' so that it is
 available for this Directory Proxy Server? (yes / no) [yes]:
Enter the DN of an account on ldap-east-01.example.com:389 with which to
 create or manage the 'cn=Proxy
User' account [cn=Directory Manager]:

Enter the password for 'cn=Directory Manager':

Created 'cn=Proxy User,cn=Root DNs,cn=config'
Testing 'cn=Proxy User' privileges ..... Done
Verifying backend 'dc=example,dc=com' ..... Done
```

12. Repeat steps 9-12 for the servers in the other location. Then, press **Enter** to finish configuring the location.
13. Review the configuration summary. Once you have confirmed that the changes are correct, press **Enter** to write the configuration.

```
>>>> Configuration Summary

  External Server Security:  SSL
  Proxy User DN:             cn=Proxy User,cn=Root DNs,cn=config

  Location east
    Failover Order: west
    Servers: localhost:1636

  Location west
    Failover Order: east
    Servers: localhost:2636

  Base DN: dc=example,dc=com
    Servers: localhost:1636, localhost:2636


    b)  back
    q)  quit
    w)  write configuration file

Enter choice [w]:
```

14. Next, apply the configuration changes locally to the Directory Proxy Server. If you have any Server SDK extensions, make sure to run the **`manage-extension`** tool, then press **Enter** to apply the changes to the Directory Proxy Server. Alternatively, you can quit and instead run the **`dsconfig`** batch file at

a later time. Once the changes have been applied, you cannot use the **create-initial-proxy-config** tool to configure this Directory Proxy Server again. Instead, use the **dsconfig** tool.

```
This tool can apply the configuration changes to the local Identity Proxy.
 This requires any configured Server SDK extensions to be in place. Do you
 want to do
this? (yes / no) [yes]:
```

If you open the generated `proxy-cfg.txt` file or the `logs/config-audit.log` file, you will see that a configuration element hierarchy has been created: locations, health checks, external servers, load-balancing algorithms, request processors, and subtree views.

About the dsconfig configuration tool

The **dsconfig** tool is the text-based management tool used to configure the underlying Directory Server configuration.

The `dsconfig` tool has three operational modes: interactive mode, non-interactive mode, and batch mode.

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

Using dsconfig in interactive command-line mode

In interactive mode, the **dsconfig** tool offers a filtering mechanism that only displays the most common configuration elements.

About this task

The user can specify that more expert level objects and configuration properties be shown using the menu system.

Running **dsconfig** in interactive command-line mode provides a user-friendly, menu-driven interface for accessing and configuring the PingDirectoryProxy Server.

Steps

1. To start `dsconfig` in interactive command-line mode, invoke the `dsconfig` script without any arguments.
   You are prompted for connection and authentication information to the Directory Proxy Server, and then a menu displays the available operation types.
2. To accept the default values, press **Enter**.

   ⓘ **Note:**

   In some cases, a default value is provided in square brackets. For example, [389] indicates that the default value for that field is port 389.

3. To skip the connection and authentication prompts, provide the connection and authentication information using the command-line options of `dsconfig`.

Changing the dsconfig object menu

The purpose of object levels is topresent only those properties that an administrator will likely use.

About this task

Because some configuration objects are more likely to be modified than others, the PingDirectoryProxy Server provides four different object menus that hide or expose configuration objects to the user. The `object` type is a convenience feature designed to improve menu readability.

The following object menus are available:

Basic

Only includes the components that are configured most frequently.

Standard

Includes all components in the Basic menu plus other components that might occasionally need to be altered in many environments.

Advanced

Includes all components in the Basic and Standard menus plus other components that might require configuration under special circumstances or that might be harmful if configured incorrectly.

Expert

Includes all components in the Basic, Standard, and Advanced menus plus other components that almost never require configuration, or that could seriously impact the functionality of the server if not properly configured.

To change the `dsconfig` object menu:

Steps

1. Using `dsconfig`, repeat steps 1–6 in *To Install the in Interactive Mode*.
2. On the **PingDirectoryProxy Server configuration** main menu, enter the letter `o` to change the object level.

   Basic objects are displayed by default.
3. Enter a number corresponding to an object level of your choice.
   - Enter `1` for Basic.
   - Enter `2` for Standard.
   - Enter `3` for Advanced.
   - Enter `4` for Expert.
4. Review the menu at the new object level.

   Additional configuration options for the Directory Proxy Server components are displayed.

Using dsconfig in non-interactive mode

The **dsconfig** non-interactive command-line mode provides a simple way to make arbitrary changes to the Directory Proxy Server by invoking it from the command line.

Steps

1. To use administrative scripts to automate configuration changes, run the **dsconfig** command in non-interactive mode.

   Non-interactive mode is convenient for scripting applications.

   > ⓘ **Note:**
   >
   > If you plan to make changes to multiple configuration objects at the same time, then the batch mode might be more appropriate.

2. Use the **dsconfig** tool to update a single configuration object using command-line arguments to provide all of the necessary information.

   The following shows the general format for the non-interactive command line.

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

   > ⓘ **Note:**
   >
   > The `--no-prompt` argument indicates that you want to use non-interactive mode. The `{sub-command}` is used to indicate which general action to perform. The `{globalArgs}` argument provides a set of arguments that specify how to connect and authenticate to the Directory Proxy Server. Global arguments can be standard LDAP connection parameters or SASL connection parameters depending on your setup. For example, using standard LDAP connections, you can invoke the **dsconfig** tool, as shown.
   >
   > ```
   > $ bin/dsconfig --no-prompt list-backends \
   >   --hostname server.example.com \
   >   --port 389 \
   >   --bindDN uid=admin,dc=example,dc=com \
   >   --bindPassword password
   > ```

3. If your system uses SASL GSSAPI (Kerberos), invoke **dsconfig** as shown.

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

4. To always display the advanced properties, use the `--advanced` command-line option.

   > ⓘ **Note:**
   >
   > The `{subcommandArgs}` argument contains a set of arguments specific to the particular subcommand that you want to invoke.
   >
   > Global arguments can appear anywhere on the command line, including before the subcommand and after or intermingled with subcommand-specific arguments. The subcommand-specific arguments can appear anywhere after the subcommand.

Getting the equivalent dsconfig non-interactive mode command

Steps

1. Use **dsconfig** in interactive mode to make changes to a configuration, but do not apply the changes (that is, do not enter the letter `f`).
2. To view the equivalent non-interactive command, enter `d`.
3. View the equivalent command, and then press **Enter** to continue.

   Based on an example in the previous section, changes made to the `db-cache-percent` returns the following.

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

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

Using dsconfig batch mode

Configure the Directory Proxy Server in `dsconfig` batch mode.

About this task

The PingDirectoryProxy Server provides a **dsconfig** batching mechanism that reads multiple **dsconfig** invocations from a file and executes them sequentially.

The batch file provides advantages over standard scripting by minimizing LDAP connections and Java virtual machine (JVM) invocations that normally occur with each **dsconfig** call. Batch mode is the best method to use with setup scripts when moving from a development environment to test environment or from a test environment to a production environment. The `--no-prompt` option is required with **dsconfig** in batch mode.

If a **dsconfig** command has a missing or incorrect argument, the command fails and aborts the batch process without applying any changes to the Directory Proxy Server. The **dsconfig** command supports a **--batch-continue-on-error** option that instructs **dsconfig** to apply all changes and skip any errors.

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

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

Steps

1. Create a text file that lists each **dsconfig** command with the complete set of properties that you want to apply to the Directory Proxy Server.

   > ⓘ **Note:**
   >
   > The items in this file should be in the same format as those accepted by the **dsconfig** command.
   >
   > The batch file can have blank lines for spacing and lines starting with a pound sign (#) for comments.
   >
   > The batch file also supports a "\" line continuation character for long commands that require multiple lines.

   ```
   # This dsconfig operation creates the exAccountNumber global attribute
    index.
   ```

```
dsconfig create-global-attribute-index
--processor-name ou_people_dc_example_dc_com-eb-req-processor
--index-name exAccountNumber --set prime-index:true

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

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

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

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

2. To read and execute the commands, run **`dsconfig`** with the `--batch-file` option.

Using PingDirectory Server or PingDirectoryProxy Server with PingFederate OAuth tokens

Configure an access token validator to use PingFederate OAuth tokens with PingDirectory.

Before you begin

You need the following information:

- The runtime engine service port of the PingFederate server, usually 9031
- The client ID of an OAuth 2.0 client configured on the PingFederate server
- The client secret of the OAuth 2.0 client
- The IP address of the PingFederate server

About this task

After you configure a PingFederate server to issue OAuth 2 tokens, you must make these tokens compatible with SCIM 2.0 operations on PingDirectory Server or PingDirectoryProxy Server.

This section explains how to set up an access token validator to handle this task.

Steps

1. Register the PingFederate server using the following command.

```
dsconfig create-external-server \
  --server-name PingFederateInstance \
  --type http \
  --set base-url:https://<PingFed IP address>:<PingFed port> \
  --set hostname-verification-method:allow-all \
```

```
  --set "trust-manager-provider:Blind Trust"
```

> ⓘ **Note:**
>
> In this example, the hostname verification method is set to allow-all and the Blind Trust manager provider is used for the sake of simplicity. You should not use these settings for production environments.

2. Create the access token validator using the following command.

```
dsconfig create-access-token-validator \
  --validator-name PingFederateValidator \
  --type ping-federate \
  --set enabled:true \
  --set authorization-server:PingFederateInstance \
  --set client-id:client-id \
  --set client-secret:client-secret
```

> ⓘ **Note:**
>
> Take the *client-id* and *client-secret* values from the PingFederate OAuth 2 client that will be used with the PingDirectory Server or PingDirectoryProxy Server.

3. Add the access token validator to the SCIM 2 HTTP Servlet configuration with the following command.

```
dsconfig set-http-servlet-extension-prop \
  --extension-name SCIM2 \
  --set access-token-validator:PingFederateValidator
```

4. Test the validator by sending a GET request to `/scim/v2/ServiceProviderConfig`.

   This endpoint does not require any scopes to access, just a valid bearer token. The sever should return a response similar to the following.

```
{
    "schemas": [
        "urn:ietf:params:scim:schemas:core:2.0:ServiceProviderConfig"
    ],
    "patch": {
        "supported": true
    },
    "bulk": {
        "supported": false,
        "maxOperations": 0,
        "maxPayloadSize": 0
    },
    "filter": {
        "supported": true,
        "maxResults": 0
    },
    "changePassword": {
        "supported": true
    },
    "sort": {
        "supported": false
    },
    "etag": {
        "supported": false
    },
    "authenticationSchemes": [
```

```
        {
            "name": "OAuth 2.0 Bearer Token",
            "description": "The OAuth 2.0 Bearer Token Authentication
 scheme. OAuth enables clients to access protected resources by obtaining
 an access token, which is defined in RFC 6750 as \"a string representing
 an access authorization issued to the client\", rather than using the
 resource owner's credentials directly.",
            "specUri": "http://tools.ietf.org/html/rfc6750",
            "type": "oauthbearertoken",
            "primary": true
        }
    ],
    "meta": {
        "resourceType": "ServiceProviderConfig",
        "location": "https://localhost:8443/scim/v2/ServiceProviderConfig"
    }
}
```

Topology configuration

Topology configuration enables automatic server grouping and configuration change mirroring. It uses a master and slave architecture for mirroring shared data across the topology.

All writes and updates are forwarded to the master, which forwards them to all other servers. Reads can be served by any server in the group, and servers can be added to an existing topology at installation.

> ⓘ **Note:**
>
> To remove a server from the topology, you must uninstall it with the uninstall tool.

Topology master requirements and selection

A topology master server receives configuration changes from other servers in the topology, verifies the changes, and then makes the changes available to all connected servers.

When updating, the master sends a digest of its subtree contents. If the node's digest differs from the master's, the server node knows it is not synchronized. The servers pull the entire subtree from the master if they detect that they are not synchronized.

A server detects it is not synchronized with the master under the following conditions:

- The server's subtree digest differs from the master's digest at the end of its periodic polling interval.
- One or more servers have been added to or removed from the topology.

The master of the topology is selected by prioritizing servers based on:

- Minimum supported product version
- Availability
- Server version
- Earliest start time
- Startup UUID (smaller is preferred)

After determining a master, the topology data is reviewed from all available servers, every five seconds by default, to determine if any new information identifies a better server master. If a new server can be the master and no other servers have indicated that they should be the master, it will communicate its eligibility to the other servers. This ensures that all servers accept the same master at approximately the same time, within a few milliseconds of each other. If there is no better master, the initial master maintains the role.

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

- A majority of servers is reachable from that master.

  The master server itself is considered while determining this majority.

- There is only a single master in the entire topology.

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

Topology components
When you install a server , you can add it to an existing topology, cloning the server's configuration. Topology settings are designed to operate without additional configuration. If required, some settings can be adjusted to fit the needs of the environment.

Server configuration settings

Configuration settings for the topology are configured in the global configuration and in the config file handler backend. They are topology settings, but they are unique to each server and are not mirrored. Settings must be kept the same on all servers.

The global configuration object contains a single topology setting, force-as-master-for-mirrored-data. This should be set to `true` on only one of the servers in the topology, and is used only if the topology cannot determine a master because most of the servers are not available. A server with this setting enabled is assigned the role of master if no suitable master can be determined.

The config file handler backend defines three topology **`mirrored-subtree`** settings:

mirrored-subtree-peer-polling-interval

Specifies the frequency at which the server polls its topology peers to identify any changes warranting a new master selection. A lower value ensures a faster failover, but it also causes more traffic among the peers. The default value is 5 seconds. If no suitable master is found, the polling frequency adjusts to 100 milliseconds until a new master is selected.

mirrored-subtree-entry-update-timeout

Specifies the maximum length of time to wait for an entry update operation, such as add, delete, modify or modify-dn, to be applied by the master on all of the servers in the topology. The default is 10 seconds, but updates can take up to twice as long if master selection is in progress at the time the update operation is received.

mirrored-subtree-search-timeout

Specifies the maximum length of time in milliseconds to wait for search operations to complete. The default is 10 seconds.

Topology settings

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

Monitor data for the topology
Each server has a monitor that exposes that server's view of the topology in its monitor backend, so that peer servers can periodically read this information to identify changes in the topology.

Topology data includes the following:

- The server ID of the current master, if the master is not known
- The instance name of the current master or if a master is not set, a description stating why a master is not set
- A flag indicating which server thinks that it should be the master

- A flag indicating which server is the current master
- A flag indicating a server that was forced as master
- The total number of configured peers in the topology group
- The peers connected to this server
- The current availability of this server
- A flag indicating that a server is not synchronized with its master or another node in the topology if the master is unknown
- The amount of time in milliseconds that multiple masters were detected by this server
- The amount of time in milliseconds that no suitable server is found to act as master
- A SHA-256 digest encoded as a base-64 string for the current subtree contents

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

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

Using the Configuration API

PingDirectoryProxy Server provides a Configuration API when updating the server configuration with LDAP is not possible. The API is consistent with the System for Cross-domain Identity Management (SCIM) 2.0 protocol and uses JSON as a text exchange format, so all request headers allow the application/json content type.

About this task
The server includes a servlet extension that provides read and write access to the server's configuration over HTTP.

Steps

- To add the extension to one of the server's HTTP Connection Handlers, run the following code.

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

> ⓘ **Note:**
>
> By default, the extension is enabled for new installations. You can enable the extension for existing deployments.

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

Authentication and authorization with the Configuration API

Use this topic for how to make changes for customizing authentication and authorization access with the Configuration API.

Authentication

Clients must use HTTP basic authentication to authenticate to the Configuration API. If the username value is not a distinguished name (DN), then it resolves to a DN value using the identity mapper associated with the Configuration servlet. By default, the Configuration API uses an identity mapper that allows an entry's UID value to be used as a username. To customize this behavior, either customize the default identity mapper or specify a different identity mapper using the Configuration servlet's `identity-mapper` property. The following code provides an example.

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

Authorization

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

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

The Configuration API and the dsconfig tool relationship

The Configuration API is designed to mirror the **`dsconfig`** tool, using the same names for properties and object types.

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

The API includes the filter, sort, and pagination query parameters described by the System for Cross-domain Identity Management (SCIM) specification. Request specific attributes using the attributes query parameter, whose value must be a comma-delimited list of properties to be returned, such as `attributes=baseDN,description`. You can exclude attributes from responses by specifying the excludedAttributes parameter.

Configuration API operations supported in REST APIs

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

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

ⓘ **Note:**

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

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

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

GET example

This topic provides an example of a GET request and response concerning the userRoot backend for reference.

GET request

The following code example is a GET request for information about the **`userRoot`** backend.

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

GET response

The following code example is the response.

```
{
  "schemas": [
    "urn:unboundid:schemas:configuration:2.0:backend:local-db"
  ],
```

```
"id": "userRoot",
"meta": {
  "resourceType": "Local DB Backend",
  "location": "http://localhost:5033/config/backends/userRoot"
},
"backendID": "userRoot2",
"backgroundPrime": "false",
"backupFilePermissions": "700",
"baseDN": [
  "dc=example2,dc=com"
],
"checkpointOnCloseCount": "2",
"cleanerThreadWaitTime": "120000",
"compressEntries": "false",
"continuePrimeAfterCacheFull": "false",
"dbBackgroundSyncInterval": "1 s",
"dbCachePercent": "10",
"dbCacheSize": "0 b",
"dbCheckpointerBytesInterval": "20 mb",
"dbCheckpointerHighPriority": "false",
"dbCheckpointerWakeupInterval": "1 m",
"dbCleanOnExplicitGC": "false",
"dbCleanerMinUtilization": "75",
"dbCompactKeyPrefixes": "true",
"dbDirectory": "db",
"dbDirectoryPermissions": "700",
"dbEvictorCriticalPercentage": "0",
"dbEvictorLruOnly": "false",
"dbEvictorNodesPerScan": "10",
"dbFileCacheSize": "1000",
"dbImportCachePercent": "60",
"dbLogFileMax": "50 mb",
"dbLoggingFileHandlerOn": "true",
"dbLoggingLevel": "CONFIG",
"dbNumCleanerThreads": "0",
"dbNumLockTables": "0",
"dbRunCleaner": "true",
"dbTxnNoSync": "false",
"dbTxnWriteNoSync": "true",
"dbUseThreadLocalHandles": "true",
"deadlockRetryLimit": "10",
"defaultCacheMode": "cache-keys-and-values",
"defaultTxnMaxLockTimeout": "10 s",
"defaultTxnMinLockTimeout": "10 s",
"enabled": "false",
"explodedIndexEntryThreshold": "4000",
"exportThreadCount": "0",
"externalTxnDefaultBackendLockBehavior": "acquire-before-retries",
"externalTxnDefaultMaxLockTimeout": "100 ms",
"externalTxnDefaultMinLockTimeout": "100 ms",
"externalTxnDefaultRetryAttempts": "2",
"hashEntries": "false",
"id2childrenIndexEntryLimit": "66",
"importTempDirectory": "import-tmp",
"importThreadCount": "16",
"indexEntryLimit": "4000",
"isPrivateBackend": "false",
"javaClass": "com.unboundid.directory.server.backends.jeb.BackendImpl",
"jeProperty": [
  "je.cleaner.adjustUtilization=false",
  "je.nodeMaxEntries=32"
],
"numRecentChanges": "50000",
"offlineProcessDatabaseOpenTimeout": "1 h",
```

```
   "primeAllIndexes": "true",
   "primeMethod": [
     "none"
   ],
   "primeThreadCount": "2",
   "primeTimeLimit": "0 ms",
   "processFiltersWithUndefinedAttributeTypes": "false",
   "returnUnavailableForUntrustedIndex": "true",
   "returnUnavailableWhenDisabled": "true",
   "setDegradedAlertForUntrustedIndex": "true",
   "setDegradedAlertWhenDisabled": "true",
   "subtreeDeleteBatchSize": "5000",
   "subtreeDeleteSizeLimit": "5000",
   "uncachedId2entryCacheMode": "cache-keys-only",
   "writabilityMode": "enabled"
}
```

GET list example

See the following example of a GET request and response for all local backends for reference.

GET request

The following is a code example GET request for all local backends.

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

GET response

The following is a code example GET response, which is shortened.

```
{
  "schemas": [
    "urn:ietf:params:scim:api:messages:2.0:ListResponse"
  ],
  "totalResults": 24,
  "Resources": [
    {
      "schemas": [
        "urn:unboundid:schemas:configuration:2.0:backend:ldif"
      ],
      "id": "adminRoot",
      "meta": {
        "resourceType": "LDIF Backend",
        "location": "http://localhost:5033/config/backends/adminRoot"
      },
      "backendID": "adminRoot",
      "backupFilePermissions": "700",
      "baseDN": [
        "cn=topology,cn=config"
      ],
      "enabled": "true",
      "isPrivateBackend": "true",
      "javaClass":
 "com.unboundid.directory.server.backends.LDIFBackend",
      "ldifFile": "config/admin-backend.ldif",
      "returnUnavailableWhenDisabled": "true",
      "setDegradedAlertWhenDisabled": "false",
      "writabilityMode": "enabled"
    },
    {
      "schemas": [
```

```
        "urn:unboundid:schemas:configuration:2.0:backend:trust-store"
      ],
      "id": "ads-truststore",
      "meta": {
        "resourceType": "Trust Store Backend",
        "location": "http://localhost:5033/config/backends/ads-
truststore"
      },
      "backendID": "ads-truststore",
      "backupFilePermissions": "700",
      "baseDN": [
        "cn=ads-truststore"
      ],
      "enabled": "true",
      "javaClass":
 "com.unboundid.directory.server.backends.TrustStoreBackend",
      "returnUnavailableWhenDisabled": "true",
      "setDegradedAlertWhenDisabled": "true",
      "trustStoreFile": "config/server.keystore",
      "trustStorePin": "********",
      "trustStoreType": "JKS",
      "writabilityMode": "enabled"
    },
    {
      "schemas": [
        "urn:unboundid:schemas:configuration:2.0:backend:alarm"
      ],
      "id": "alarms",
      "meta": {
        "resourceType": "Alarm Backend",
        "location": "http://localhost:5033/config/backends/alarms"
      },
  ...
```

PATCH example
Modify the Configuration API using the HTTP PATCH method.

The PATCH request body is a JSON object formatted according to the System for Cross-domain Identity Management (SCIM) patch request. The Configuration API supports a subset of possible values for the path attribute that indicates the configuration attribute to modify.

Modify the configuration object's attributes according to the information in the following table, which details the comparable `dsconfig modify-[object]` options to each PATCH request.

Operations and PATCH requests to modify the configuration object's attributes

Operation	PATCH request	Comparable `dsconfig modify-[object]` options
Set the single-valued `description` attribute to a new value.	`{ "op" : "replace", "path" : "description", "value" : "A new backend." }`	`$ dsconfig set-backend-prop --backend-name userRoot \ --set "description:A new backend"`
Add a new value to the multi-valued `jeProperty` attribute.	`{ "op" : "add", "path" : "jeProperty", "value" : "je.env.backgroundReadLi`	`$ dsconfig set-backend-prop --backend-name userRoot \ --add je-property:je.env.backgroundReadLimi`
Remove a value from a multi-valued property. In this case, path specifies a SCIM filter identifying the value to remove.	`{ "op" : "remove", "path" : "[jeProperty eq \"je.cleaner.adjustUtili \"]" }`	`$ dsconfig set-backend-prop --backend-name userRoot \ --remove je-property:je.cleaner.adjustUtilizat`
Second operation to remove a value from a multi-valued property, where the `path` specifies both an attribute to modify and a SCIM filter whose attribute is the following value.	`{ "op" : "remove", "path" : "jeProperty[value eq \"je.nodeMaxEntries=32\"]`	`$ dsconfig set-backend-prop --backend-name userRoot \ --remove je-property:je.nodeMaxEntries=32`
Option to remove one or more values of a multi-valued attribute. This has the effect of restoring the attribute's value to its default value.	`{ "op" : "remove", "path" : "id2childrenIndexEntryLi }`	`$ dsconfig set-backend-prop --backend-name userRoot \ --reset id2childrenIndexEntryLimit`

The following is the full example request.

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

{
  "schemas" :
 [ "urn:ietf:params:scim:api:messages:2.0:PatchOp" ],
  "Operations" : [ {
    "op" : "replace",
    "path" : "description",
    "value" : "A new backend."
```

```
  }, {
    "op" : "add",
    "path" : "jeProperty",
    "value" : "je.env.backgroundReadLimit=0"
  }, {
    "op" : "remove",
    "path" : "[jeProperty eq
 \"je.cleaner.adjustUtilization=false\"]"
  }, {
    "op" : "remove",
    "path" : "jeProperty[value eq \"je.nodeMaxEntries=32\"]"
  }, {
    "op" : "remove",
    "path" : "id2childrenIndexEntryLimit"
  } ]
}
```

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

```
{
 "schemas": [
   "urn:unboundid:schemas:configuration:2.0:backend:local-db"
 ],
 "id": "userRoot2",
 "meta": {
   "resourceType": "Local DB Backend",
   "location": "http://example.com:5033/config/backends/
userRoot2"
 },
 "backendID": "userRoot2",
 "backgroundPrime": "false",
 "backupFilePermissions": "700",
 "baseDN": [
   "dc=example2,dc=com"
 ],
 "checkpointOnCloseCount": "2",
 "cleanerThreadWaitTime": "120000",
 "compressEntries": "false",
 "continuePrimeAfterCacheFull": "false",
 "dbBackgroundSyncInterval": "1 s",
 "dbCachePercent": "10",
 "dbCacheSize": "0 b",
 "dbCheckpointerBytesInterval": "20 mb",
 "dbCheckpointerHighPriority": "false",
 "dbCheckpointerWakeupInterval": "1 m",
 "dbCleanOnExplicitGC": "false",
 "dbCleanerMinUtilization": "75",
 "dbCompactKeyPrefixes": "true",
 "dbDirectory": "db",
 "dbDirectoryPermissions": "700",
 "dbEvictorCriticalPercentage": "0",
 "dbEvictorLruOnly": "false",
 "dbEvictorNodesPerScan": "10",
 "dbFileCacheSize": "1000",
 "dbImportCachePercent": "60",
 "dbLogFileMax": "50 mb",
 "dbLoggingFileHandlerOn": "true",
 "dbLoggingLevel": "CONFIG",
 "dbNumCleanerThreads": "0",
```

```
 "dbNumLockTables": "0",
 "dbRunCleaner": "true",
 "dbTxnNoSync": "false",
 "dbTxnWriteNoSync": "true",
 "dbUseThreadLocalHandles": "true",
 "deadlockRetryLimit": "10",
 "defaultCacheMode": "cache-keys-and-values",
 "defaultTxnMaxLockTimeout": "10 s",
 "defaultTxnMinLockTimeout": "10 s",
 "description": "123",  "enabled": "false",
 "explodedIndexEntryThreshold": "4000",
 "exportThreadCount": "0",
 "externalTxnDefaultBackendLockBehavior": "acquire-before-
retries",
 "externalTxnDefaultMaxLockTimeout": "100 ms",
 "externalTxnDefaultMinLockTimeout": "100 ms",
 "externalTxnDefaultRetryAttempts": "2",
 "hashEntries": "false",
 "importTempDirectory": "import-tmp",
 "importThreadCount": "16",
 "indexEntryLimit": "4000",
 "isPrivateBackend": "false",
 "javaClass":
 "com.unboundid.directory.server.backends.jeb.BackendImpl",
 "jeProperty": [  "\"je.env.backgroundReadLimit=0\""
 ],
 "numRecentChanges": "50000",
 "offlineProcessDatabaseOpenTimeout": "1 h",
 "primeAllIndexes": "true",
 "primeMethod": [
   "none"
 ],
 "primeThreadCount": "2",
 "primeTimeLimit": "0 ms",
 "processFiltersWithUndefinedAttributeTypes": "false",
 "returnUnavailableForUntrustedIndex": "true",
 "returnUnavailableWhenDisabled": "true",
 "setDegradedAlertForUntrustedIndex": "true",
 "setDegradedAlertWhenDisabled": "true",
 "subtreeDeleteBatchSize": "5000",
 "subtreeDeleteSizeLimit": "5000",
 "uncachedId2entryCacheMode": "cache-keys-only",
 "writabilityMode": "enabled",
 "urn:unboundid:schemas:configuration:messages:2.0": {
   "requiredActions": [
     {
       "property": "jeProperty",
       "type": "componentRestart",
       "synopsis": "In order for this modification to take effect,
          the component must be restarted, either by disabling and
          re-enabling it, or by restarting the server"
     },
     {
       "property": "id2childrenIndexEntryLimit",
       "type": "other",
       "synopsis": "If this limit is increased, then the contents
          of the backend must be exported to LDIF and re-imported
 to
          allow the new limit to be used for any id2children keys
          that had already hit the previous limit."
     }
   ]
 }
```

```
}
```

Configuration API paths

The Configuration API and supported sub-paths are available under the `/config` path.

A full listing of supported sub-paths is available when you access the base `/config/ResourceTypes` endpoint.

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

Some paths reflect hierarchical relationships between objects. For example, properties of a local DB VLV index for the `userRoot` backend are available using a path like `/config/backends/userRoot/local-db-indexes/uid`. Some paths represent singleton objects, which have properties but cannot be deleted or created. These paths can be differentiated from others by their singular, rather than plural, relation name, such as `global-configuration`.

The following sample response is abbreviated.

```
{
  "schemas": [
    "urn:ietf:params:scim:api:messages:2.0:ListResponse"
  ],
  "totalResults": 520,
  "Resources": [
    {
      "schemas": [

 "urn:ietf:params:scim:schemas:core:2.0:ResourceType"
      ],
      "id": "dsee-compat-access-control-handler",
      "name": "DSEE Compat Access Control Handler",
      "description": "The DSEE Compat Access Control
              Handler provides an implementation that uses
 syntax
              compatible with the Sun Java System Directory
 Server
              Enterprise Edition access control handler.",
      "endpoint": "/access-control-handler",
      "meta": {
        "resourceType": "ResourceType",
        "location": "http://example.com:5033/config/
ResourceTypes/dsee-compat-access-control-handler"
      }
    },
    {
      "schemas": [

 "urn:ietf:params:scim:schemas:core:2.0:ResourceType"
      ],
      "id": "access-control-handler",
      "name": "Access Control Handler",
      "description": "Access Control Handlers manage the
              application-wide access control. The server's
 access
              control handler is defined through an extensible
              interface, so that alternate implementations can
 be created.
```

```
               Only one access control handler may be active in
 the server
                          at any given time.",
      "endpoint": "/access-control-handler",
      "meta": {
        "resourceType": "ResourceType",
        "location": "http://example.com:5033/config/
ResourceTypes/access-control-handler"
      }
    },
    {
 ...
```

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

Sort and filter objects

The Configuration API supports System for Cross-domain Identity Management (SCIM) parameters for filter, sorting, and pagination.

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

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

Update properties

The Configuration API supports the HTTP PUT method as an alternative to modifying objects with HTTP PATCH.

With PUT, the server computes the differences between the object in the request with the current version in the server and performs modifications where necessary. The server never removes attributes that are not specified in the request. The API responds with the entire modified object.

Sample request:

```
PUT /config/backends/userRoot
Host: example.com:5033
Accept: application/scim+json
{
  "description" : "A new description."
```

```
}
```

Sample response:

```
{
  "schemas": [
    "urn:unboundid:schemas:configuration:2.0:backend:local-db"
  ],
  "id": "userRoot",
  "meta": {
    "resourceType": "Local DB Backend",
    "location": "http://example.com:5033/config/backends/
userRoot"
  },
  "backendID": "userRoot",
  "backgroundPrime": "false",
  "backupFilePermissions": "700",
  "baseDN": [
    "dc=example,dc=com"
  ],
  "checkpointOnCloseCount": "2",
  "cleanerThreadWaitTime": "120000",
  "compressEntries": "false",
  "continuePrimeAfterCacheFull": "false",
  "dbBackgroundSyncInterval": "1 s",
  "dbCachePercent": "25",
  "dbCacheSize": "0 b",
  "dbCheckpointerBytesInterval": "20 mb",
  "dbCheckpointerHighPriority": "false",
  "dbCheckpointerWakeupInterval": "30 s",
  "dbCleanOnExplicitGC": "false",
  "dbCleanerMinUtilization": "75",
  "dbCompactKeyPrefixes": "true",
  "dbDirectory": "db",
  "dbDirectoryPermissions": "700",
  "dbEvictorCriticalPercentage": "5",
  "dbEvictorLruOnly": "false",
  "dbEvictorNodesPerScan": "10",
  "dbFileCacheSize": "1000",
  "dbImportCachePercent": "60",
  "dbLogFileMax": "50 mb",
  "dbLoggingFileHandlerOn": "true",
  "dbLoggingLevel": "CONFIG",
  "dbNumCleanerThreads": "1",
  "dbNumLockTables": "0",
  "dbRunCleaner": "true",
  "dbTxnNoSync": "false",
  "dbTxnWriteNoSync": "true",
  "dbUseThreadLocalHandles": "true",
  "deadlockRetryLimit": "10",
  "defaultCacheMode":
  "cache-keys-and-values",
  "defaultTxnMaxLockTimeout": "10 s",
  "defaultTxnMinLockTimeout": "10 s",
  "description": "abc",
  "enabled": "true",
  "explodedIndexEntryThreshold": "4000",
  "exportThreadCount": "0",
  "externalTxnDefaultBackendLockBehavior":
  "acquire-before-retries",
  "externalTxnDefaultMaxLockTimeout": "100 ms",
  "externalTxnDefaultMinLockTimeout": "100 ms",
  "externalTxnDefaultRetryAttempts": "2",
```

```
  "hashEntries": "true",
  "importTempDirectory": "import-tmp",
  "importThreadCount": "16",
  "indexEntryLimit": "4000",
  "isPrivateBackend": "false",
  "javaClass":
 "com.unboundid.directory.server.backends.jeb.BackendImpl",
  "numRecentChanges":
 "50000",  "offlineProcessDatabaseOpenTimeout": "1 h",
  "primeAllIndexes": "true",
  "primeMethod": [
    "none"
  ],
  "primeThreadCount": "2",
  "primeTimeLimit": "0 ms",
  "processFiltersWithUndefinedAttributeTypes": "false",
  "returnUnavailableForUntrustedIndex": "true",
  "returnUnavailableWhenDisabled": "true",
  "setDegradedAlertForUntrustedIndex": "true",
  "setDegradedAlertWhenDisabled": "true",
  "subtreeDeleteBatchSize": "5000",
  "subtreeDeleteSizeLimit": "100000",
  "uncachedId2entryCacheMode": "cache-keys-only",
  "writabilityMode": "enabled"
}
```

Administrative actions

Updating a property might require an administrative action before changes can take effect.

If you need administrative actions to update a property, the server returns `200 Success`. Any actions are returned in the `urn:unboundid:schemas:configuration:messages:2.0` section of the JSON response that represents the entire object that was created or modified.

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

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

```
  }
```

Updating servers and server groups
You can configure servers as part of a server group so that configuration changes applied to a single server are applied to all servers in a group.

When managing a server that is a member of a server group, creating or updating objects using the Configuration API requires the `applyChangeTo` query attribute. The behavior and acceptable values for this parameter are identical to the `dsconfig` parameter of the same name. A value of `single-server` or `server-group` can be specified.

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

Configuration API responses
Clients of the API should examine the HTTP response code to determine the success or failure of a request.

The following are response codes and their meanings.

Response Code	Description	Response Body
200 Success	The requested operation succeeded, with the response body being the configuration object that was created or modified. If further actions are required, they are included in the `urn:unboundid:schemas:configuration:messages:2.0` object.	List of objects, object properties, or administrative actions
204 No Content	The requested operation succeeded and no further information has been provided, as in the case of a DELETE operation.	None
400 Bad Request	The request contents are incorrectly formatted or a request is made for an invalid API version.	Error summary and optional message
401 Unauthorized	User authentication is required. Some user agents, such as browsers, might respond by prompting for credentials. If the request specified credentials in an Authorization header, they are invalid.	None
403 Forbidden	The requested operation is forbidden, either because the user does not have sufficient privileges or some other constraint, such as an object is edit-only and cannot be deleted.	None
404 Not Found	The requested path does not refer to an existing object or object relation.	Error summary and optional message

Response Code	Description	Response Body
409 Conflict	The requested operation could not be performed due to the current state of the configuration. For example, an attempt was made to create an object that already exists, or an attempt was made to delete an object that is referenced by another object.	Error summary and optional message
415 Unsupported Media Type	The request is such that the Accept header does not indicate that JSON is an acceptable format for a response.	None
500 Server Error	The server encountered an unexpected error. Report server errors to Customer Support.	Error summary and optional message

ⓘ **Note:**

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

The following is an example encoded error message.

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

Working with the Directory REST API

The Directory REST API is the native interface for client access to the PingDirectoryProxy Server. Instead of trying to manage directory hierarchy or require attribute mapping, the Directory REST API provides direct access to directory data in a way that is dynamic, discoverable, and efficient.

Before you begin

The Directory REST API gives developers who are more comfortable with REST than LDAP access to arbitrary directory data in a way that ensures directory data remains consistent regardless of whether it is accessed from LDAP or REST. The Directory API is enabled during server setup. After setup, individual services and applications can be enabled or disabled by configuring the HTTPS Connection Handler.

While both the Directory REST API and System for Cross-domain Identity Management (SCIM) provide REST access to directory data, the goals of the two protocols are different. SCIM is useful to generic, external clients that require simple, narrow access to identity data, but because it is a less common standard for identity stores, it might not offer as much functionality or be as user-friendly as the Directory REST API.

The Directory REST API can be used for the following operations.

HTTP operation	Resource endpoint	Description	Allowed query parameters
DELETE	**/directory/v1/{dn}**	Delete an entry.	
GET	**/directory/v1**	Get metadata about the API and server.	
GET	**/directory/v1/{dn}**	Retrieve a single entry.	▪ expand ▪ includeAttributes ▪ excludeAttributes
GET	**/directory/v1/{dn}/subtree**	Search an entry's descendants.	▪ filter ▪ searchScope ▪ cursor ▪ limit ▪ includeAttributes ▪ excludeAttributes
GET	**/directory/v1/schemas**	Retrieve the schemas of all available object classes.	
GET	**/directory/v1/schemas/{objectclass}**	Retrieve schema for a specific object class.	
GET	**/directory/v1/schemas/_operationalAttributes**	Retrieve schema for operational attributes.	
GET	**/directory/v1/me**	Alias for retrieving the current user.	
PATCH	**/directory/v1/{dn}**	Modify an entry (add or delete values).	expand
POST	**/directory/v1**	Create a new entry.	expand
PUT	**/directory/v1/{dn}**	Modify or rename an entry.	expand

Steps

- Configure the Directory REST API with any of the following properties using **dsconfig**:

Command	Description
basic-auth-enabled	Specifies whether users can connect to the service with HTTP Basic authentication. If disabled, users need a bearer token. If changed, the server must be restarted, or any HTTP Connection Handlers referencing this service disabled and re-enabled. Basic authentication is enabled by default.
identity-mapper	If HTTP Basic authentication is enabled, the identity mapper referenced by this distinguished name (DN) must be used to map the

Command	Description
	user names provided to user entries. By default, an identity mapper is provided, which maps a fully-qualified DN to an entry. For changes to take effect, the server must be restarted, or any HTTP connection handlers referencing this service disabled and re-enabled.
access-token-validator	Specifies the subset of this server's Access Token Validators (by DN), which can validate Bearer authentication tokens. By default, if no validators are specified, then any of the validators on the server can be used. For changes to take effect, the server must be restarted, or any HTTP Connection Handlers referencing this service disabled and re-enabled.
access-token-scope	The scope that must be present in Bearer tokens in order to be accepted by this service. If no value is provided, Bearer token authentication is disabled, and only Basic authentication is used. By default, no value is provided. Changes to this value take effect immediately.
audience	A string or URI audience that must be present in Bearer tokens in order to be accepted by this service. If no value is provided, any audience is acceptable. By default, no value is provided. Changes to this value take effect immediately.
max-page-size	The maximum number of entries to be returned in one page from the search endpoint. Actual results returned might be lower due to the limit query parameter on the request and the actual number of available results. The value must be an integer between 1 and 1000. The default value is 100. Changes to this value take effect immediately.
schemas-endpoint-objectclass	The list of object classes that will be returned by the **/schemas/** endpoint in the REST API. By default, no schemas are returned. Changes to this value take effect immediately.

The following example uses **dsconfig** to configure an **objectClass** entity.

```
dsconfig set-http-servlet-extension-props --extension-name "Directory REST
 API" \
  --add schemas-endpoint-objectclass:ubidPerson
```

Generating a summary of configuration components

The `config-diff` toolgenerates a summary of the configuration in a local or remote directory server instance.

About this task

The `config-diff` tool is useful for comparing configuration settings on the directory server instance when troubleshooting issues or when verifying configuration settings on newly-added servers. The tool can interact with the local configuration regardless of whether the server is running.

Steps

- To generate a summary of configuration components, run `config-diff`
  $ bin/config-diff generates a summary of a local online server.

- To generate a comparison of the current configuration with the pre-installation configuration, run the following.

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

  This comparison ignores any changes that could be made by the installer and writes the output to a file called `configuration-steps.dsconfig`.

  You can use this file as the basis for a script to configure a new server identical to the local server.
- To view other available tool options, run `config-diff --help`.

Configuring server groups

The PingDirectoryProxy Server provides a mechanism for setting up administrative domains that synchronize configuration changes among servers in a server group.

About this task

After you have set up a server group, you can make an update on one server using **`dsconfig`**, then apply the change to the other servers in the group using the `--applyChangeTo server-group` option of the **`dsconfig`** non-interactive command. If you want to apply the change to one server in the group, use the `--applyChangeTo single-server` option. When using **`dsconfig`** in interactive command-line mode, you are asked if you want to apply the change to a single server or to all servers in the server group.

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

The following example procedure adds three directory server instances into the server group labeled "group-one".

Steps

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

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

2. Add any directory server to the server group.

   If you have set up replication between a set of servers, these server entries are created by the **`dsreplication`** `enable` command.

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

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

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

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

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

4. Test the server group.

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

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

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

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

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

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

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

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

   The first directory server instance should be disabled.

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

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

9. View the property on the second directory server instance.

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

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

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

DNS caching

You can use two global configuration properties to control the caching of host name-to-numeric IP address or DNS lookup results returned from the name resolution services of the underlying operating system.

About this task

Steps

1. To configure these properties, use the **`dsconfig`** tool.

   Global configuration properties and their descriptions

Global configuration property	Description
network-address-cache-ttl	Sets the Java system property **`networkaddress.cache.ttl`**, and controls the length of time in seconds that a host name-to-IP address mapping can be cached. By default, it keeps resolution results for one hour (3600 seconds). This setting applies to the server and all extensions loaded by the server.
network-address-outage-cache-enabled	Caches host name-to-IP address results in the event of a DNS outage. By default, this is set to true, meaning name resolution results are cached. Unexpected service interruptions might occur during planned or unplanned maintenance, network outages, or an infrastructure attack. This cache can allow the server to function during a DNS outage with minimal impact. This cache is not available to server extensions.

2. To reduce delays due to unnecessary DNS lookups, follow these recommendations:
   a. Maintain a connection pool in the client app rather than opening new connections for each bind.
   b. Add appropriate records to DNS, including PTR records.
   c. Add `options timeout:1` or `options single-request` in the `/etc/resolv.conf` file.
   d. If IPv6 requests are causing issues, add `-Djava.net.preferIPv4Stack=true` to the `start-server.java-args` line in PingDirectory Server's `config/java.properties` file, so that running `bin/dsjavaproperties` and restarting the server no longer issues IPv6 PTR requests.

IP address reverse name lookups

The PingDirectory Server performs numeric IP address-to-host name lookups.

Numeric IP address-to-host name lookups

The following are some of the numeric IP address-to-host name lookups:

- Binding to the Directory: Decoding, examining, or evaluating a DNS bind rule
- Logging: Logging information to certain monitors or writing to the error log
- Java Management Extension (JMX): Creating a server socket
- Key Management: Generating a trust store
- Replication Server: Creating an SSL socket
- Replication Session Management: Obtaining a session or performing a handshake with a replication server
- Simple Authentication and Security Layer (SASL) Authentication: Applying configuration changes
- SMTP Alert Handler: Initializing or sending an alert notification

Configuring address masks

Address masks configured in Access control instructions (ACIs), connection handlers, connection criteria, and certificate handshake processing might trigger implicit reverse name lookups.

For more information about how address masks are configured in the server, see the following information for each server:

- ACI DNS: bind rules in *Managing Access Control* on page 566 for Directory Server and Directory Proxy Server
- `ds-auth-allowed-address`: *Restricting access through operational attributes in user entities* for Directory Server
- Connection Criteria: *Restricting server access based on client IP address* on page 399 for Directory Server and Directory Proxy Server
- Connection Handlers: *Restricting server access using the connection handlers* on page 399 for a configuration reference guide for all servers

Configuring traffic through a load balancer

Configure the PingDirectoryProxy Server to get traffic through a load balancer and to record the actual client's IP address.

Before you begin

By default, when a PingDirectoryProxy Server is sitting behind an intermediate HTTP server, such as a load balancer, a reverse proxy, or a cache, it logs incoming requests as originating with the intermediate HTTP server instead of the client that sent the request.

Steps

1. To record the actual client's IP address to the trace log, enable **`X-Forwarded-*`** handling in both the intermediate HTTP server and the PingDirectoryProxy Server.
2. Edit the appropriate connection handler object, HTTPS or HTTP, and set **`use-forwarded-headers`** to **`true`**.

   > ⓘ **Note:**
   >
   > When **`use-forwarded-headers`** is set to **`true`**, the server uses the client IP address and port information in the **`X-Forwarded-*`** headers instead of the address and port of the entity that's sending the request, the load balancer. This client address information shows up in logs, such as in the **`from`** field of the HTTP REQUEST and HTTP RESPONSE messages.

3. On the load balancer, configure settings to provide the **`X-Forwarded-*`** information, such as **`X-Forwarded-Host:`**.

Managing root user accounts

PingDirectoryProxy Server provides a default root user, `cn=Directory Manager`, that is stored in the server's configuration file, such as under `cn=Root DNs,cn=config`.

About this task

The root user is the LDAP-equivalent of a UNIX superuser account and inherits its read-write privileges from the default root privilege set.

Steps

- To create or update root users, use the `dscconfig` tool.

```
bin/dsconfig create-root-dn-user --user-name "Joanne Smith" \
  --set last-name:Smith \
  --set first-name:Joanne \
  --set user-id:jsmith \
  --set 'email-address:jsmith@example.com' \
```

```
  --set mobile-telephone-number:8889997777 \
  --set home-telephone-number:5556667777 \
  --set work-telephone-number:4445556666
```

> ⓘ **Note:**
>
> Root user entries are stored in the server's configuration.

- To limit full access to all of Directory Proxy Server, create separate administrator accounts with limited privileges so that you can identify the administrator responsible for a particular change.

> ⓘ **Note:**
>
> Separate user accounts for each administrator make it possible to enable password policy functionality, such as password expiration, password history, and requiring secure authentication, for each administrator.

Default root privileges

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

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

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

Default Root Privileges

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

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

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

Other Available Privileges

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

Configuring locations

PingDirectoryProxy Server defines locations, both for the LDAP external servers and the proxy server instances themselves. A location defines a collection of servers that share access and latency characteristics. For example, your deployment might include two data centers, one in the east and one in the west. These data centers would be configured as two locations in the Directory Proxy Server. Each location is associated with a name and an ordered list of failover locations, which could be used if none of the servers in the preferred location are available. You can define these locations using the Administrative Console or the command line.

The Directory Proxy Server itself is also associated with a location. This location is specified in the global configuration properties of the Directory Proxy Server. If the load balancing algorithm's `use-location` property is set to true, then the load balancing component of the Directory Proxy Server refers to the Directory Proxy Server's location to determine the external servers it prefers to communicate with.

Configuring locations using dsconfig

Steps

1. Use the **`dsconfig`** tool to configure the LDAP external server locations.

```
$ bin/dsconfig
```

2. Type the host name or IP address for your Directory Proxy Server, or press **Enter** to accept the default, localhost.

```
Directory Proxy Server host name or IP address [localhost]:
```

3. Type the number corresponding how you want to connect to the Directory Proxy Server, or press **Enter** to accept the default, LDAP.

```
How do you want to connect?
  1) LDAP
  2) LDAP with SSL
  3) LDAP with StartTLS
```

4. Type the port number for your Directory Proxy Server, or press **Enter** to accept the default, 389.

```
Directory Proxy Server port number [389]:
```

5. Type the administrator's bind DN or press **Enter** to accept the default (cn=Directory Manager), and then type the password.

```
Administrator user bind DN [cn=Directory Manager]:
Password for user 'cn=Directory Manager':
```

6. In the Directory Proxy Server main menu, enter the number corresponding to location configuration. Then, enter the number to create a new location.
7. Enter the name of the new location. This example demonstrates configuring a location called East. Enter **f** to finish configuring the location. Repeat this procedure to create a location called West.

```
>>>>  Enter a name for the location that you want to create: east

>>>>  Configure the properties of the location

      Property                    Value(s)
      ----------------------------------
   1) description                  -
   2) preferred-failover-location  -

   ?) help
   f) finish - create the new location
   d) display the equivalent dsconfig arguments to
```

```
      create this object
   b) back
   q) quit

Enter choice [b]: f
```

8. Next, edit the configuration of an existing location, in this example a location named East.

```
>>>> Location menu
What would you like to do?

  1) List existing locations
  2) Create a new location
  3) View and edit an existing location
  4) Delete an existing location

  b) back
  q) quit

Enter choice [b]: 3

>>>> Select the location from the following list:
   1) East
   2) West

   b) back
   q) quit

Enter choice [b]: 1
```

9. Define the preferred failover location property for East. This property provides alternate locations that can be used if servers in this location are not available. If more than one location is provided, the Directory Proxy Server tries the locations in the order listed.

```
 >>>> Configure the properties of the Location

     Property                    Value(s)
     ----------------------------------
  1) description                 -
  2) preferred-failover-location  -

  ?) help
  f) finish - create the new location
  d) display the equivalent dsconfig arguments to create this object
  b) back
  q) quit

Enter choice [b]: 2

...

Do you want to modify the 'preferred-failover-location' property?

   1) Add one or more values

   ?) help
   q) quit

Enter choice [1]: 2

Select the locations you wish to add:

  1) East
  2) West
```

```
  3) Create a new location
  4) Add all locations

...

Enter one or more choices separated by commas[b]: 2
```

10. Verify and apply your change to the property.

```
Do you want to modify the 'preferred-failover-location' property?

  1) Use the value: West
  2) Add one or more values
  3) Remove one or more values
  4) Leave undefined
  5) Revert changes

  ?) help
  q) quit

Enter choice [1]:

>>>> Configure the properties of the location

     Property                    Value(s)
     ----------------------------------
  1) description                 -
  2) preferred-failover-location  West

  ?) help
  f) finish - apply any changes to the Location
  d) display the equivalent dsconfig command lines to either
     re-create this object or only to apply pending changes
  b) back
  q) quit

Enter choice [b]: f
```

11. Repeat steps 8 and 9 for the West location, assigning it a failover location of East.

Modifying locations using dsconfig

Steps

1. Use the **`dsconfig`** tool to configure the LDAP external server locations.

```
$ bin/dsconfig
```

2. Type the host name or IP address for your Directory Proxy Server, or press **Enter** to accept the default, localhost.

```
Directory Proxy Server host name or IP address [localhost]:
```

3. Type the number corresponding how you want to connect to the Directory Proxy Server, or press **Enter** to accept the default, LDAP.

```
How do you want to connect?
  1) LDAP
  2) LDAP with SSL
  3) LDAP with StartTLS
```

4. Type the port number for your Directory Proxy Server, or press **Enter** to accept the default, 389.

```
Directory Proxy Server port number [389]:
```

5. Type the administrator's bind DN or press **Enter** to accept the default (cn=Directory Manager), and then type the password.

```
Administrator user bind DN [cn=Directory Manager]:
Password for user 'cn=Directory Manager':
```

6. In the Directory Proxy Server main menu, enter the number corresponding to Global Configuration. Then enter the number to view and edit the Global Configuration.
7. Enter the number associated with the location configuration property.

```
Enter choice [b]: 2
```

8. Specify a new location for this Directory Proxy Server instance, in this example the East location. Operations involving communications with other servers may prefer servers in the same location to ensure low-latency responses.

```
>>>> Configuring the 'location' property

...

Do you want to modify the 'location' property?

  1) Leave undefined
  2) Change it to the location: East
  3) Change it to the location: West
  4) Create a new location

  b) back
  q) quit

Enter choice [b]: 2
```

9. Enter **f** to finish the operation.

```
Enter choice [b]: f
```

Configuring batched transactions

About this task

You can configure the Directory Proxy Server to use batched transactions in both simple and entry-balanced configurations. The batched transactions feature supports two implementations: the standard LDAP transactions per RFC 5805 and the PingDirectoryProxy Server proprietary implementation, known as the *multi-update extended operation*. Batched transactions can be used through the Directory Proxy Server in both simple and entry-balanced configurations although only in cases in which all operations within the transaction request may be processed within the same backend server and within the same Berkeley DB JE backend. Batched transactions cannot be processed across multiple servers or multiple Directory Server backends.

The multi-update extended operation makes it possible to submit multiple updates in a single request. These updates may be processed either as individual operations or as a single atomic unit. When the Directory Proxy Server receives a Start Batched Transaction request, it will queue all associated operations in memory until the End Batched Transaction request is received with the intention to commit, at which point the set of operations is sent as a single multi-update extended request to the Directory Server.

Add, delete, modify, modify DN, and password modify extended operations may be included in the set of operations processed during a batch transaction. The operations are processed sequentially in the order in which they were included in the extended request. If an error occurs while processing an operation in the set, then the server can be instructed to continue the processing or to cancel any remaining operations. If the operations are not cancelled, you can configure the server to process all operations as a single atomic unit.

Because of this use of multi-update, the external Directory Server must be configured to allow multi-update extended requests made by the Directory Proxy Server on behalf of the DN submitting the batched transaction. For example, the following Directory Server **dsconfig** command grants anonymous access to the multi-update extended request. The submitter of the request still needs access rights for the individual operations within the multiple-update.

```
$ bin/dsconfig set-access-control-handler-prop \
  --add 'global-aci:(extop="1.3.6.1.4.1.30221.2.6.17")(version 3.0;
 acl "Anonymous access to multi-update extended request"; allow (read)
 userdn="ldap:///anyone";)'
```

Batched transactions are managed by the Batched Transactions Extended Operation Handler. You can use it to configure the start transaction and end transaction operations used to indicate the set of add, delete, modify, modify DN, and/or password modify operations as a single atomic unit.

Steps

1. You can configure batched transactions using the `dsconfig` command as follows:

   ```
   $ bin/dsconfig set-extended-operation-handler-prop \
     --handler-name "Batched Transactions" \
     --set enabled:true
   ```
2. Configure the external servers to allow the multi-update extended operation by granting access rights to the feature. See example in the previous section.

Configuring server health checks

You can use the PingDirectoryProxy Server to configure different types of health checks for your deployment. The health checks define external server availability as either being available, unavailable, or degraded. The external server health is given a value from 0 to 10, which is used to determine if the server is available and how that server compares to other servers with the same state. Load-balancing algorithms can be used to check the score and prefer servers with higher scores over those with lower scores.

An individual health check can be defined for use against all external servers or assigned to individual external servers, as determined by the `use-for-all-servers` parameter within the health check configuration object. If `use-for-all-servers` is set to true, the Directory Proxy Server applies the health check to all external servers in all locations. If `use-for-all-servers` is set to false, then the health check is only employed against an external server if the configuration object for that external server lists the health check.

For more information about health checks and the type of health checks supported by PingDirectoryProxy Server, see *About LDAP Health Checks*.

About the default health checks

By default, the Directory Proxy Server has two health check instances enabled for use on all servers:

- **Consume Admin Alerts**. This health check detects administrative alerts from the Directory Server, as soon as they are issued, by maintaining an LDAP persistent search for changes within the `cn=alerts` branch of the Directory Server. When the Directory Proxy Server is notified by the Directory Server of a new alert, it immediately retrieves the base `cn=monitor` entry of the Directory Server. If this entry has a value for the `unavailable-alert-type` attribute, then the Directory Proxy Server will consider it unavailable. If this entry has a value for the `degraded-alert-type` attribute, then the Directory Proxy Server will consider it degraded.
- **Get Root DSE**. This health check detects if the root DSE entry exists on the LDAP external server. As this entry always exists on a PingDirectory Server, the absence of the entry suggests that the LDAP external server may be degraded or unavailable.

About creating a custom health check

You can create a new health check from scratch or use an existing health check as a template for the configuration of a new health check. If you choose to create a custom health check, you can create one of the following types:

- **Admin Alert Health Check**. This health check watches for administrative alerts generated by the LDAP external server to determine whether the server has entered a degraded or unavailable state.
- **Groovy Scripted LDAP Health Check**. This health check allows you to create custom LDAP health checks in a dynamically-loaded Groovy script, which implements the `ScriptedLDAPHealthCheck` class defined in the Server SDK.
- **Replication Backlog Health Check**. While the Admin Alert Health Check consumes replication backlog alerts emitted from external servers, a finer definition of external server health based on replication backlog can be defined with this health check. If a server falls too far behind in replication, then the Directory Proxy Server can stop sending requests to it. A server is classified as degraded or unavailable if the threshold is reached for the number of backlogged changes, the age of the oldest backlogged change, or both.
- **Search LDAP Health Check**. This health check performs searches on an LDAP external server and gauges the health of the server depending if the expected results were returned within an acceptable response time. For example, if an error occurs while attempting to communicate with the server, then the server is considered unavailable. You can also apply filters to the results to use values within the monitor entry as indicators of server health.
- **Third Party LDAP Health Check**. This health check allows you to define LDAP health check implementations in third-party code using the Server SDK.
- **Work Queue Busyness Health Check**. This health check may be used to monitor the percentage of time that worker threads in backend servers spend processing requests.

Configuring a health check using dsconfig

Steps

1. Use the **`dsconfig`** tool to configure the LDAP external server locations.

   ```
   $ bin/dsconfig
   ```

2. Type the host name or IP address for your Directory Proxy Server, or press **Enter** to accept the default, localhost.

   ```
   Directory Proxy Server host name or IP address [localhost]:
   ```

3. Type the number corresponding how you want to connect to the Directory Proxy Server, or press **Enter** to accept the default, LDAP.

   ```
   How do you want to connect?
     1) LDAP
     2) LDAP with SSL
     3) LDAP with StartTLS
   ```

4. Type the port number for your Directory Proxy Server, or press **Enter** to accept the default, 389.

   ```
   Directory Proxy Server port number [389]:
   ```

5. Type the administrator's bind DN or press **Enter** to accept the default (cn=Directory Manager), and then type the password.

   ```
   Administrator user bind DN [cn=Directory Manager]:
   Password for user 'cn=Directory Manager':
   ```

6. In the Directory Proxy Server main menu, enter the number corresponding to LDAP health checks. Enter the number to create a new LDAP Health Check, then press `n` to create a new health check from scratch.

7. Select the type of health check you want to create. This example demonstrates the creation of a new search LDAP health check.

```
>>> Select the type of LDAP Health Check that you want to create:

      1)  Admin Alert LDAP Health Check
      2)  Custom LDAP Health Check
      3)  Groovy Scripted LDAP Health Check
      4)  Replication Backlog LDAP Health Check
      5)  Search LDAP Health Check
      6)  Third Party LDAP Health Check
      7)  Work Queue Busyness LDAP Health Check

      ?) help
      c) cancel
      q) quit

Enter choice [c]: 5
```

8. Specify a name for the new health check. In this example, the health check is named `Get example.com`.

```
>>>> Enter a name for the search LDAP Health Check that you want to create:
 Get example.com
```

9. Enable the new health check.

```
>>>> Configuring the 'enabled' property

Indicates whether this LDAP health check is enabled for use in the server.

Select a value for the 'enabled' property:

   1) true
   2) false

   ?) help
   c) cancel
   q) quit

Enter choice [c]: 1
```

10. Next, configure the properties of the health check. You may need to modify the `base-dn` property, as well as one or more response time thresholds for non-local external servers, accommodating WAN latency. Below is a Search LDAP Health Check for the single entry `dc=example,dc=com`, which allows non-local responses of up to 2 seconds to still be considered healthy.

```
>>>> Configure the properties of the Search LDAP Health Check

           Property                                    Value(s)
           ------------------------------------------------------------
      1)   description                                 -
      2)   enabled                                     true
      3)   use-for-all-servers                         false
      4)   base-dn                                     "dc=example,dc=com"
      5)   scope                                       base-object
      6)   filter                                      (objectClass=*)
      7)   maximum-local-available-response-time       1 s
      8)   maximum-nonlocal-available-response-time    2 s
      9)   minimum-local-degraded-response-time        500 ms
      10)  minimum-nonlocal-degraded-response-time     1 s
      11)  maximum-local-degraded-response-time        10 s
      12)  maximum-nonlocal-degraded-response-time     10 s
      13)  minimum-local-unavailable-response-time     5 s
```

```
      14)  minimum-nonlocal-unavailable-response-time 5 s
      15)  allow-no-entries-returned                  true
      16)  allow-multiple-entries-returned            true
      17)  available-filter                           -
      18)  degraded-filter                            -
      19)  unavailable-filter                         -

      ?) help
      f) finish - create the new Search LDAP Health Check
      d) display the equivalent dsconfig arguments to create this object
      b) back
      q) quit
```

Configuring LDAP external servers

The LDAP external server configuration element defines the connection, location, and health check information necessary for the Directory Proxy Server to communicate with the server properly.

PingDirectoryProxy Server includes a tool, **`prepare-external-server`**, for configuring communication between the Directory Proxy Server and the LDAP backend server. After you add a new LDAP external server to an existing installation, we strongly recommend that you run this tool to automatically create the user account necessary for communications. The **`prepare-external-server`** tool does not make configuration changes to the local Directory Proxy Server, only the external server is modified. When you run this tool, you must supply the user account and password that you specified for the Directory Proxy Server during configuration, `cn=Proxy User` by default.

ⓘ **Important:** You should not use `cn=Directory Manager` as the account to use for communication between the Directory Proxy Server and the Directory Server. For security reasons, the account used to communicate between the Directory Proxy Server and the Directory Server should not be directly accessible by clients accessing the Directory Proxy Server. The account that you choose should meet the following criteria:

- For all server types, it should not exist in the Directory Proxy Server but only in the backend directory server instances.
- For Ping Identity Directory Server, this user should be a root user.
- For Ping Identity Directory Server, this user should not automatically inherit the default set of root privileges, but instead should have exactly the following set of privileges: `bypass-read-acl`, `config-read`, `lockdown-mode`, `proxied-auth`, and stream-values.
- For Sun Directory Servers, the account should be created below the `cn=Root DNs,cn=config` entry and the `nsSizeLimit`, `nsTimeLimit`, `nsLookThroughLimit`, and `nsIdleTimeout` values for the account should be set to -1. You also need to create access control rules to grant the user account appropriate permissions within the server. The **`prepare-external-server`** tool handles all of this work automatically.

About the prepare-external-server tool

Use the **`prepare-external-server`** tool if you have added LDAP external servers using **`dsconfig`**. The **`create-initial-proxy-config`** tool automatically runs the **`prepare-external-server`** tool to configure server communications so that you do not need to invoke it separately. The **`create-initial-proxy-config`** tool verifies that the proxy user account exists and has the correct password and required privileges. If it detects any problems, it prompts for manager credentials to rectify them.

If you want the **`prepare-external-server`** tool to add the LDAP external server's certificates to the Directory Proxy Server's trust store, you must include the `--proxyTrustStorePath` option, and either the `--proxyTrustStorePassword` or the `--proxyTrustStorePasswordFile` option. The default location of the Directory Proxy Server trust store is `config/truststore`. The pin is encoded in the `config/truststore.pin` file.

For example, run the tool as follows to prepare a PingDirectory Server on the remote host, ds-east-01.example.com, listening on port 1389 for access by the Directory Proxy Server using the default user account `cn=Proxy User`:

```
prepare-external-server --hostname ds-east-01.example.com \
--port 1389 --baseDN dc=example,dc=com --proxyBindPassword secret
```

When the **`prepare-external-server`** command above is executed, it creates the `cn=Proxy User` Root DN entry as well as an access control rule in the Directory Server to grant the proxy user the proxy access right.

ⓘ **Note:** For non-Ping Identity servers, the `--baseDN` argument is required for the **`prepare-external-server`** tool. The base DN is used to create the global ACI entries for these servers.

Configuring server communication using the prepare-external-server tool

About this task

The following example illustrates how to run the **`prepare-external-server`** tool to prepare a Directory Server on the remote host, ds-east-01.example.com, listening on port 1636. The Directory Server is being accessed by a Directory Proxy Server that uses the default user account `cn=Proxy User,cn=Root DNs,cn=config`. Since a password to the truststore is not provided, the truststore defined in the **`--proxyTrustStorePath`** is referenced in a read-only manner.

Steps

- Use the **`prepare-external-server`** tool to prepare the Directory Server. Follow the prompts to set up the external server.

```
$ ./PingDirectoryProxy/bin/prepare-external-server \
  --baseDN dc=example,dc=com
  --proxyBindPassword password \
  --hostname ds-east-01.example.com \
  --useSSL \
  --port 1636
  --proxyTrustStorePath /full/path/to/trust/store \
  --proxyTrustStorePassword secret
```

```
Testing connection to ds-east-01.example.com:1636 .....

Do you wish to trust the following certificate?

Certificate Subject: CN=ds-east-01.example.com, O=Example Self-Signed
 Certificate
Issuer Subject:      CN=ds-east-01.example.com, O=Example Self-Signed
 Certificate
Validity:            Thu May 21 08:02:30 CDT 2009 to Wed May 16 08:02:30 CDT
 2029

Enter 'y' to trust the certificate or 'n' to reject it.

y

The certificate was added to the local trust store

Done
Testing 'cn=Proxy User' access to ds-east-01.example.com:1636 ..... Failed
 to bind as
'cn=Proxy User'
```

```
Would you like to create or modify root user 'cn=Proxy User' so that it is
 available
for this Directory Proxy Server? (yes / no) [yes]:

Enter the DN of an account on ds-east-01.example.com:1636 with which to
 create or
manage the 'cn=Proxy User' account [cn=Directory Manager]:

Enter the password for 'cn=Directory Manager':

Created 'cn=Proxy User,cn=Root DNs,cn=config'

Testing 'cn=Proxy User' privileges ..... Done
```

Configuring an external server using dsconfig

Steps

1. Use the **dsconfig** tool to create and configure external servers. Then, specify the host name, connection method, port number, and bind DN as described in previous procedures.

   ```
   $ bin/dsconfig
   ```
2. In the Directory Proxy Server main menu, enter the number corresponding to external servers. Then, enter the number to create a new external server.
3. Select the type of server you want to create. This example creates a new Ping Identity Directory Server.

   ```
   >>>> Select the type of external server that you want to create:

     1) Ping Identity DS external server
     2) JDBC external server
     3) LDAP external server
     4) Sun DS external server

     ?) help
     c) cancel
     q) quit

   Enter choice [c]: 1
   ```
4. Specify a name for the new external server. In this example, the external server is named east1.

   ```
   >>>> Enter a name for the Ping Identity DS external server that you want
   to create: east1
   ```
5. Configure the host name or IP address of the target LDAP external server.

   ```
   Enter a value for the 'server-host-name' property:east1.example.com
   ```
6. Next, configure the location property of the new external server.

   ```
   Do you want to modify the 'location' property?

      1) Leave undefined
      2) Change it to the location: East
      3) Change it to the location: West
      4) Create a new location

      ?) help
      q) quit

   Enter choice [1]: 2
   ```

7. Next, define the bind DN and bind password.

```
Do you want to modify the 'bind-dn' property?

   1) Leave undefined
   2) Change the value

   ?) help
   q) quit

Enter choice [1]: 2

Enter a value for the 'bind-dn' property [continue]: cn=Proxy User,cn=Root
 DNs,cn=config

Enter choice [b]: 6

...

Do you want to modify the 'password' property?

   1) Leave undefined
   2) Change the value
   ?) help
   q) quit

Enter choice [1]: 2

Enter a value for the 'password' property [continue]:
Confirm the value for the 'password' property:
```

8. Enter `f` to finish the operation.

```
Enter choice [b]: f

The Ping Identity DS external server was created successfully.
```

   Once you have completed adding the server, run the **`prepare-external-server`** tool to configure communications between the Directory Proxy Server and the Ping Identity Directory Server(s).

Configuring authentication with a SASL external certificate

About this task

By default, the Directory Proxy Server authenticates to the Directory Server using LDAP simple authentication (with a bind DN and a password). However, the Directory Proxy Server can be configured to use SASL EXTERNAL to authenticate to the Directory Server with a client certificate.

Have Directory Proxy Server instances installed and configured to communicate with the backend Directory Server instances using either SSL or StartTLS. After the servers are configured, perform the following steps to configure SASL EXTERNAL authentication.

Steps

1. Create a JKS keystore that includes a public and private key pair for a certificate that the Directory Proxy Server instance(s) will use to authenticate to the Directory Server instance(s). Run the following command in the instance root of one of the Directory Proxy Server instances. When prompted for a keystore password, enter a strong password to protect the certificate. When prompted for the key password, press **ENTER** to use the keystore password to protect the private key:

```
$ keytool -genkeypair \
  -keystore config/proxy-user-keystore \
```

```
  -storetype JKS \
  -keyalg RSA \
  -keysize 2048 \
  -alias proxy-user-cert \
  -dname "cn=Proxy User,cn=Root DNs,cn=config" \
  -validity 7300
```

2. Create a **config/proxy-user-keystore.pin** file that contains a single line that is the keystore password provided in the previous step.
3. If there are other Directory Proxy Server instances in the topology, copy the **proxy-user-keystore** and **proxy-user-keystore.pin** files into the **config** directory for all instances.
4. Use the following command to export the public component of the proxy user certificate to a text file:

```
$ keytool -export \
  -keystore config/proxy-user-keystore \
  -alias proxy-user-cert \
  -file config/proxy-user-cert.txt
```

5. Copy the **proxy-user-cert.txt** file into the **config** directory of all Directory Server instances. Import that certificate into each server's primary trust store by running the following command from the server root. When prompted for the keystore password, enter the password contained in the **config/truststore.pin** file. When prompted to trust the certificate, enter **yes**.

```
$ keytool -import \
  -keystore config/truststore \
  -alias proxy-user-cert \
  -file config/proxy-user-cert.txt
```

6. Update the configuration for each Directory Proxy Server instance to create a new key manager provider that will obtain its certificate from the **config/proxy-user-keystore** file. Run the following **dsconfig** command:

```
$ dsconfig create-key-manager-provider \
  --provider-name "Proxy User Certificate" \
  --type file-based \
  --set enabled:true \
  --set key-store-file:config/proxy-user-keystore \
  --set key-store-type:JKS \
  --set key-store-pin-file:config/proxy-user-keystore.pin
```

7. Update the configuration for each LDAP external server in each Directory Proxy Server instance to use the newly-created key manager provider, and also to use SASL EXTERNAL authentication instead of LDAP simple authentication. Run the following **dsconfig** command:

```
$ dsconfig set-external-server-prop \
  --server-name ds1.example.com:636 \
  --set authentication-method:external \
  --set "key-manager-provider:Proxy User Certificate"
```

Results

After these changes, the Directory Proxy Server should re-establish connections to the LDAP external server and authenticate with SASL EXTERNAL. Verify that the Directory Proxy Server is still able to communicate with all backend servers by running the **bin/status** command. All of the servers listed in the "--- LDAP External Servers ---" section should be available. Review the Directory Server access log can to make sure that the BIND RESULT log messages used to authenticate the connections from the Directory Proxy Server include **authType="SASL"**, **saslMechanism="EXTERNAL"**, **resultCode=0**, and **authDN="cn=Proxy User,cn=Root DNs,cn=config"**.

Servers and certificates

The following provides a detailed description of PingDirectory Server certificate types.

PingDirectory Server uses two main types of certificates:

Listener certificates

Used to secure communication through TLS.

Inter-server certificates

Used to authenticate between server instances in a topology.

Listener certificates

When a client initiates TLS negotiation with the server, the server presents a certificate chain to the client and the certificate at the head of the chain functions as a listener certificate.

Because the client decides whether to trust the certificate chain, it is recommended that the chain be signed by an issuer whom the client is likely to trust or that the client can be easily configured to trust.

You can create self-signed certificates with long lifespans, but a certificate that a certification authority signs is likely to have a relatively short lifespan. Commercial authorities typically issue certificates that are valid for only one or two years, but some authorities use shorter validity windows.

Short certificate lifespans offer some security benefits. In particular, because most clients do not verify whether a certificate has been revoked, a shorter validity window minimizes the timeframe that a compromised certificate can be used. If the process for replacing certificates is streamlined or automated, administrative inconvenience can be kept to a minimum.

Listener certificates are stored in key stores that are referenced by key manager providers, which in turn provide the logic and configuration for accessing the key stores. If a server component, like a connection handler, requires access to a certificate that it presents to a peer during the TLS negotiation process, that component must reference the key manager provider that points to the key store containing the appropriate certificate. If the key store contains multiple certificates, and if the component referencing the key store includes a property specifying the certificate's nickname, the certificate with that alias is selected. Otherwise, the server lets the Java virtual machine (JVM) select a certificate that might not be well-defined.

The server also provides trust manager providers, which determine whether to trust the certificate chains with which it is presented. A trust manager provider can reference a specified trust store file, but other options include the JVM default trust store, which uses the Java installation's default set of trusted issuers, and the blind trust manager provider, which automatically trusts every certificate chain that is presented to it.

> ⓘ **Note:**
>
> Never use a blind trust manager in a production environment because it leaves the server vulnerable to impersonation and man-in-the-middle attacks. However, a blind trust manager can be convenient in test environments when troubleshooting certain types of problems.

Replacing listener certificates

Certificate authorities typically restrict the lifespans of the certificates that they sign. If you use a certification authority to issue listener certificates, you are likely replacing the certificates on a regular basis.

About this task

The **`replace-certificate`** tool performs the following steps:

1. Obtain a new certificate chain.
2. Make necessary updates to the key manager provider and the connection handler configurations
3. Update the server instance listener configuration with the new certificate.

The **replace-certificate** tool offers the following modes of operation:

Interactive mode

Walks you through the process of obtaining a new certificate and installing it in the server. Interactive mode also displays the non-interactive commands that are required to achieve the same result.

Non-interactive mode

Useful when scripting the process of replacing a certificate.

Steps

- To replace a listener certificate, run the **replace-listener-certificate** subcommand of the **replace-certificate** tool.

> ⓘ **Note:**
>
> You can replace certificates manually, but the **replace-certificate** tool automates the process. The **replace-certificate** tool provides information about multiple listener certificates during the transitional phase that occurs when you install them.
>
> The **replace-listener-certificate** subcommand takes arguments that provide the following information:
>
> - Arguments required to authenticate to PingDirectoryProxy Server, such as --bindDN and --bindPasswordFile
> - Details about the key store that contains the new certificate
> - Updates that must be made to the key and trust manager providers
> - Whether to signal the HTTP connection handler to reload its certificates after the update is complete

The following arguments are available:

Argument	Description
--source-key-store-file {path}	Path to the Java KeyStore (JKS) or PKCS #12 file that contains the private key entry with the new certificate chain. This argument is required.
--source-key-store-password {password}	Clear-text password that is needed to access the contents of the source key store.
--source-key-store-password-file {path}	Path to the file that contains the password necessary to access the contents of the source key store. The file can contain the password in the clear or can be encrypted with a definition from the server's encryption settings database.
--source-certificate-alias {alias}	Password that is required to access the appropriate private key in the source key store. If neither the --source-private-key-password nor the --source-private-key-password-file argument is provided, the key store password is used as the private key password.
--source-private-key-password-file {path}	Path to the file that contains the password needed to access the appropriate private key in the source key store. The file can contain the password in the clear or can be encrypted with

Argument	Description
	a definition from the server's encryption settings database. If neither the --source-private-key-password nor the --source-private-key-password-file argument is provided, the key store password is used as the private key password.
--key-manager-provider {name}	Name of the key manager provider that is updated to use the new certificate chain. The value must identify a file-based key manager provider, and the new certificate chain must be enabled. Defaults to JKS if a value is not specified.
--trust-manager-provider {name}	Name of the trust manager provider that is updated with the information required to trust the new certificate chain. The value must identify a file-based trust manager provider, and the new certificate chain must be enabled. If neither this argument nor the --use-jvm-default-trust-manager-provider argument is provided, the tool assumes that the name of the trust manager provider is identical to the name of the key manager provider.
--use-jvm-default-trust-manager-provider	Indicates that the server must be configured to use the JVM-default trust manager provider, which trusts certificates signed by issuers in the `cacerts` trust store provided with the Java virtual machine (JVM), rather than updating an existing trust manager provider.
--target-certificate-alias {alias}	Alias to use for the new certificate in the key manager provider's key store, and for appropriate updates in the trust manager provider's trust store. Defaults to an alias of server-cert if a value is not specified. ⓘ **Note:** If the key manager provider's key store, or the trust manager provider's trust store, already contains an entry with the given alias, the existing entry is renamed.
--reload-http-connection-handler-certificates	Indicates that the tool is requesting that the server cause any HTTPS-based connection handlers to reload their certificates, so that the connection handlers can use the updated certificate. LDAP connection handlers react to the change immediately and start presenting the new certificate chain during subsequent TLS negotiations. HTTPS connection handlers continue using the former certificate until the connection handler is restarted or until the connection handler is asked specifically to reload its certificates.

Argument	Description
	ⓘ **Note:** This option might prevent clients with existing TLS sessions that were negotiated with the former certificate from being resumed.

- To remove earlier certificates from the server instance listener configuration, run the purge-retired-listener-certificates subcommand.

ⓘ **Note:**

The purge-retired-listener-certificates subcommand does not take arguments other than the ones that are required to authenticate to the server.

By default, the **`replace-certificate`** tool updates the server instance listener configuration object to include the new listener certificate, and it merges the old and new certificates residing in the configuration object.

Inter-server certificates
Each server instance in a topology has an inter-server certificate that is generated during the setup process.

The inter-server certificate is not exposed to clients, so a trusted issuer does not need to sign it. Instead, the topology registry, representing a mirrored portion of the configuration with information about every PingDirectory Server instance in the environment, contains the information that each instance needs to trust the inter-server certificates for all other instances.

Inter-server certificates can be used to protect certain secrets that are shared among servers within the topology, like the secrets that are used to digitally sign log files, backups, and LDIF exports. Inter-server certificates include the encryption keys that reversible password-storage schemes use.

The inter-server certificate is generated with a long lifespan. Replace it only when you suspect that its private key is compromised.

Replacing the inter-server certificate
During the installation process, the inter-server certificate is generated with a long lifespan and does not require replacement under normal circumstances. You should replace the inter-server certificate only if you suspect that its private key is compromised.

About this task

The inter-server certificate is intended for use only between server instances within the same topology. Because it is not exposed to regular clients, the inter-server certificate does not need to be trusted.

The **`replace-certificate replace-inter-server-certificate`** command performs the following steps:

- Acquires the new inter-server certificate from a provided Java KeyStore (JKS) or PKCS #12 key store
- Makes the necessary updates to the `config/ads-truststore` file in the server key store
- Updates the server instance configuration object to include the new inter-server certificate

ⓘ **Note:**

To avoid the need to replace the inter-server certificate on a regular basis,use a self-signed certificate with a long lifespan. Each server instance must possess its own, unique inter-server certificate that satisfies the following conditions:

- Uses an RSA key pair
- Has a minimum key size of 2048 bits

The following types of certificates are not allowed:

- Certificates with an elliptic curve key pair
- Certificates with an RSA key that is smaller than 2048 bits

Steps

- To replace the inter-server certificate, run the **`replace-inter-server-certificate`** subcommand of the **`replace-certificate`**.

  The **`replace-inter-server-certificate`** subcommand takes a subset of the arguments that are used with the **`replace-listener-certificate`** subcommand, including the following arguments:

  - --source-key-store-file {path}--source-key-store-password {password}
  - --source-key-store-password-file {path}
  - --source-certificate-alias {alias}
  - --source-private-key-password {password}
  - --source-private-key-password-file {path}
- To delete earlier values that are no longer needed, run the **`purge-retired-inter-server-certificates`** subcommand.

> ⓘ **Note:**
>
> By default, the new inter-server certificate is merged with the existing values in the server instance configuration entry.

X.509 certificates
PingDirectoryProxy Server supports X.509 certificates, the most common type of certificates. *RFC 5280* describes X.509v3, which provides the current version of the specification.

An X.509v3 certificate includes the following components:

X.509 encoding version

Enables the differentiation between an X.509v3 certificate and one that conforms to an earlier or later version of the specification.

Serial number of the certificate

Integer value that uniquely identifies a certificate as issued by a certification authority.

Subject DN

Distinguished name for the certificate, which often provides details about the context in which the certificate is to be used. For more information, see *Certificate subject DNs* on page 319.

Issuer DN

Distinguished name for the issuer certificate, which is the certificate used to sign the certificate. For a self-signed certificate, this value matches the subject DN.

Validity window

Indicates the timeframe during which the certificate is considered valid. This component includes the following elements:

- notBefore

  Specifies the earliest time at which the certificate is considered valid.

- notAfter

  Specifies the latest time at which the certificate is considered valid.

Public key

Public portion of a pair of cryptographically linked keys. For more information, see *Certificate key pairs* on page 320.

Signature

A type of cryptographic proof that the certificate truly was sent from the issuer and has remained unaltered. A self-signed certificate is signed with its own private key. Otherwise, it is signed with the issuer's private key.

An X.509v3 certificate might also include the following optional components:

Subject unique ID

Uniquely identifies the certificate. This component has been deprecated in favor of the subject key identifier extension, so it is generally omitted from X.509v3 certificates.

Issuer unique ID

Subject unique ID of the issuer certificate, if available. This component has been deprecated in favor of the authority key identifier extension.

Set of extensions

Provides additional context for the certificate and the manner in which it is used. For more information, see *Certificate extensions* on page 320.

Certificate subject DNs

A certificate's subject distinguished name (DN) provides information about how the certificate should be used.

Like an LDAP DN, a certificate's subject DN consists of a comma-delimited series of attribute-value pairs. However, unlike an LDAP DN, the attribute names in a certificate subject DN are typically written in all uppercase characters.

A certificate's subject DN is also referred to as its subject. The following attributes commonly appear in a certificate subject.

Attribute name	Attribute description
`CN`	Common name **Note:** For a listener certificate, the `CN` attribute typically identifies the host name that clients use to access the certificate. However, the subject alternative name extension is recommended more highly for accomplishing the same task. Most certificate subject DNs include at least the `CN` attribute.
`E`	Email address
`OU`	Name of the organizational unit, such as the relevant department
`O`	Name of the organization or company
`L`	Name of the locality, such as the appropriate city

Attribute name	Attribute description
`ST`	Full name of the state or province
`C`	ISO 3166 country code

A certificate subject includes at least one attribute-value pair, and the `CN` attribute is typically present. Other attributes can be omitted, although the `O` and `C` attributes are also common. For example, a listener certificate for a server with an address of `ldap.example.com`, which is run by the US-based company Example Corp, might have a subject of `CN=ldap.example.com,O=Example Corp,C=US`.

Certificate key pairs
Each certificate contains a key pair that consists of two keys that are linked cryptographically. If you encrypt data with one key, the data can be only decrypted with the other key.

Although a key pair can be created easily when both keys are generated simultaneously, the process of deriving one key from the other is extremely difficult, a process categorized in cryptographic terms as computationally infeasible.

When generating a key pair, one key is designated as the public key, and the other key is designated the private key. The public key can be made widely available, but the private key must be kept secret and not shared with anyone.

As long as the secrecy of the private key is maintained, the key pair can be used to perform the following functions:

- Encryption, sometimes referred to as confidentiality

  If someone wants to send you a secret message without anyone else viewing it, the message can be encrypted with your public key. Only you possess the private key, so only you can decrypt the message.
- Digital signatures

  If you encrypt data with your private key, it can be decrypted only with your public key. Because your public key can be made widely available, this encryption method does not actually protect the content. However, digital signatures prove that a message came from you because only your private key could have generated it.

> ⓘ **Note:**
>
> When generating a digital signature, the entire message is generally not encrypted. Only a hash of the message is encrypted, typically by using a digest algorithm like SHA-256.
>
> This approach protects the integrity of a message. A decrypted signature that matches the digest of the original message guarantees that the message came from you and that it has remained unaltered since you signed it.

The following public key algorithms are used primarily in certificates that facilitate TLS communication:

- RSA, which is based on the multiplication of large prime numbers
- EC, which is based on computations that involve special types of elliptical curves

Although RSA is supported more widely than EC, it is slower and requires larger keys to achieve the same level of security. To support legacy clients, you should use an RSA certificate and choose a key size of at least 2,048 bits.

If all of your clients support EC certificates, you should use an EC certificate with a key size of at least 256 bits.

Certificate extensions
Extensions provide additional context for a certificate.

Some of the more common extension types include the following:

Subject key identifier

Holds a unique identifier for the certificate, which is generally derived from the certificate's public key.

Authority key identifier

Holds the subject key identifier for the issuer certificate. This extension type helps to identify the issuer certificate, especially when presented with an incomplete certificate chain.

Subject alternative name

Holds a list of ways that clients are expected to reference a server when establishing a connection to it.

> ⓘ **Note:**
>
> Clients must take this information into account when deciding whether to trust a server's certificate.
>
> The most common types of values include DNS names, IP addresses, and URIs. DNS names must be fully qualified, but can optionally use an asterisk in the leftmost component to match any single name in that component. For example, `*.example.com` could match `www.example.com` or `ldap.example.com`, but would not match `ldap.east.example.com` or `example.com`.

Key usage

Provides information about the manner in which the certificate is expected to be used. The following key usages are allowed:

`digitalSignature`

Indicates that the certificate can be used for digitally signing data, excluding certificates and certificate revocation lists (CRL).

`nonRepudiation`

Indicates that the certificate can be used to prevent denying the authenticity of a message. `nonRepudiation` is also known as `contentCommitment`.

`keyEncipherment`

Indicates that the certificate can be used to protect encryption keys, such as symmetric keys that are derived during TLS key agreement.

`dataEncipherment`

Indicates that the certificate can be used for encrypting data directly.

`keyAgreement`

Indicates that the certificate's public key can be used for key agreement, such as deriving the symmetric key that protects TLS communication.

`keyCertSign`

Indicates that the certificate can act as a certification authority and be used for signing other certificates.

`cRLSign`

Indicates that the certificate can be used to sign CRLs.

encipherOnly

When used in conjunction with `keyEncipherment`, indicates that the public key can be used only for encrypting data during key agreement.

decipherOnly

When used in conjunction with `keyEncipherment`, indicates that the public key can be used only for decrypting data during key agreement.

Extended key usage

Acts as an alternative to the key usage extension and provides additional high-level functionality. The following extended key usages are allowed:

serverAuth

Indicates that the server can present the certificate to the client during TLS negotiation.

clientAuth

Indicates that the client can present the certificate to the server during TLS negotiation.

codeSigning

Indicates that the certificate can be used to sign source and compiled code.

emailProtection

Indicates that the certificate can be used to sign or encrypt email messages.

timeStamping

Indicates that the certificate can be used to assert the time that an event occurred.

ocspSigning

Indicates that the certificate can be used to sign an online certificate status protocol (OCSP) response.

Basic constraints

Indicates whether the certificate can act as a certification authority and, if so, the maximum number of intermediate certificates that can follow it in a certificate chain.

Certificate chains

A certificate chain is an ordered list of one or more certificates. In such a chain, each subsequent certificate is the issuer of the previous certificate.

During TLS negotiation, the server presents a certificate chain to the client, which determines whether to trust the chain and continue with the negotiation. The client can also present its own certificate chain to the server.

If a certificate is self-signed, its chain contains only that single certificate. If a certificate is signed by a self-signed certificate authority (CA) certificate, such as a root CA, the chain contains two certificates: the server certificate and the CA certificate that follows it. If a single intermediate CA (a CA certificate that is signed by a root CA) is present, the chain contains the server certificate, followed by the intermediate CA, and then the root CA.

Intermediate certificate authorities are useful for security purposes, especially in commercial authorities. If a client trusts a root CA certificate, it is likely to trust anything with that root CA certificate at the base of its chain. Consequently, the root CA certificate must be kept secure.

ⓘ **Note:**

If the root CA certificate is compromised, any certificate that is directly or indirectly signed by it can no longer be trusted.

With intermediate CA certificates, the root certificate can be kept offline in secure storage and used only when a new intermediate CA certificate must be signed. The intermediate CA certificates can be used to sign end-entity certificates, but must be protected to avoid compromising any of the certificates. A compromised certificate must be revoked along with all of the certificates that it signed. In such a scenario, the root CA can be used to sign a new certificate.

ⓘ **Note:**

The certificate chain that the server presents to the client, or that the client presents to the server, during TLS negotiation does not always need to be the complete chain. If the root CA at the end of the chain is widely trusted, the server can assume that the client already has that root CA in its default set of trusted certificates. The server can leave that root CA off the chain with the assumption that the client will retrieve it from its default trust store. While the same assumption could theoretically be true for intermediate CA certificates, only the root CA certificate is commonly omitted. When a client receives an incomplete chain, the client looks in its default trust store to determine whether the trust store contains the issuer certificate, which it can identify by using properties like the issuer distinguished name (DN) or an authority key identifier extension.

The certificate at the head of a certificate chain, which appears as the first one in the list, is often called the end-entity certificate. If this certificate appears at the head of the chain that a server presents during TLS negotiation, it is referred to as the server certificate. If the certificate appears at the head of a chain that a client presents, it is referred to as a client certificate. The certificate at the end of a complete chain must be a root CA certificate. In the case of a self-signed certificate, the chain contains only a single certificate that serves both roles.

About representing certificates, private keys, and certificate signing requests

X.509 is an encoding format that uses the ASN.1 distinguished encoding rules (DER), which exist in binary format. When writing a certificate to a file, either a raw DER format or a plaintext format called PEM can be used.

PEM encoding consists of a line that contains the text `-----BEGIN CERTIFICATE-----`, followed by a set of lines that contains the base64-encoded representation of the raw DER bytes (typically with no more than 64 characters per line), followed by a line that contains the text `-----END CERTIFICATE-----`.

The X.509 encoding contains a certificate's public key, but not its private key. The PKCS #8 specification in *RFC 5958* describes the encoding for private keys. This approach uses a DER encoding with a PEM variant that instead uses the following header and footer, respectively.

```
 -----BEGIN PRIVATE KEY-----
-----END PRIVATE KEY-----
```

RFC 5958 also describes an encrypted representation of the private key, but that format is currently unsupported.

The PKCS #10 specification in *RFC 2986* describes the CSR format. This format uses a DER encoding with a PEM variant that uses the following header and footer, respectively.

```
-----BEGIN CERTIFICATE REQUEST-----
-----END CERTIFICATE REQUEST-----
```

Some implementations use the following alternate, nonstandard forms.

```
-----BEGIN NEW CERTIFICATE REQUEST-----
-----END NEW CERTIFICATE REQUEST-----
```

Certificate trust

When a server presents its certificate chain to a client during TLS negotiation, the client decides whether to trust the certificate chain and concludes whether it is communicating with a legitimate server instead of an impostor.

If a client is tricked through DNS hijacking into communicating with a rogue application instead of with a legitimate server, the application can steal the client's credentials, or can fool the client into concluding that it has performed an action that it has not performed. If a rogue application acts as a broker between the client and the legitimate server, the client might be unable to detect the change, and the malicious application might be capable of stealing data or altering the communication. Consequently, you should avoid `trust all` or `blind trust` options in a production environment.

When determining whether to trust a server certificate chain, a client performs the following steps.

Processing steps

1. Verifies that it has received the complete certificate chain.

   If a server presents an incomplete chain, the client must ensure that it can complete the chain with information in an explicitly provided trust store or default trust store. If the client cannot complete the certificate chain, the chain is not trusted.
2. Verifies that each subsequent certificate in the chain is the issuer certificate for, and that its private key was used to sign, the certificate that precedes it.

   > ⓘ **Note:**
   >
   > If a certificate chain contains extraneous certificates, or if a subsequent certificate did not issue the certificate that precedes it, the chain is not trusted.
3. Confirms that it has a reason to trust the certificate at the root of the chain.

   > ⓘ **Note:**
   >
   > This step is generally performed by ensuring that the root certificate authority (CA) certificate can be found in either a default trust store or a trust store that is configured for use by the client. If the client has no prior knowledge of the root CA certificate, the chain is not trusted.
4. Verifies that the current time lies within the validity window for each certificate in the chain.

   > ⓘ **Note:**
   >
   > The chain is not trusted under the following conditions:
   >
   > - When the `notBefore` value of any certificate in the chain is later than the current time.
   > - When the `notAfter` value of any certificate in the chain is earlier than the current time.

5. Verifies that the server certificate at the head of the chain is suitable for the server with which the client thinks it is communicating.

> ⓘ **Note:**
>
> The client must verify that the address used to connect to the server matches one of the following values:
>
> - The `CN` attribute of the certificate's subject.
> - One of the values of any subject alternative name extension.

These steps represent a starting point. If necessary, the client can perform additional types of validation. For example, if a root or intermediate certification authority maintains a certificate revocation list (CRL) or supports the online certificate status protocol (OCSP), the client must verify that none of the certificates in the chain has been revoked. The client can also verify that the CA certificates include the basic constraints extension, and that the server certificate does not contain too many levels. Other checks, like those that use certificate policy extensions, can also be performed.

Keystores and truststores
A keystore is a type of database that holds certificates.

The following examples represent the most common forms of keystores:

- File that uses the Java-specific Java KeyStore (JKS) format
- File that uses the standard PKCS #12 format
- Collection of files that holds certificates and private keys, typically in PEM or DER format
- Hardware security module (HSM) that makes the certificate information available through an interface like PKCS #11

PingDirectoryProxy Server supports file-based keystores by using the JKS and PKCS #12 formats and by using hardware security modules that are accessible through PKCS #11. The server does not currently support a keystore format that consists of individual certificate and private key files. To import these files into a JKS or PKCS #12 keystore, use the **`manage-certificates`** tool.

A keystore also represents a collection of entries, each of which is identified by a name that an alias calls. Keystores can have the following entry types:

Private key entries

Contain a certificate chain and a private key. When a server accepts a TLS-based connection, it uses a private key entry to obtain the certificate chain that it presents to the client. The server can also use the private key from the same entry to process its key agreement. Similarly, a client uses a private key entry when presenting its own certificate chain to a server.

Trusted certificate entries

Contain a single certificate without a private key. As the name implies, a trusted certificate entry is intended primarily for use when determining whether to trust a certificate chain that is presented during TLS negotiation.

Secret key entries

Contain a secret key only, without an associated certificate. These types of entries are not used for TLS processing. Instead, they hold symmetric encryption keys or other types of secrets.

A password, sometimes called a PIN, protects the contents of a keystore. In some cases, like with JKS keystores, some content might be accessible without a password, and a password might be required only when trying to access private keys or secret keys. In other cases, like with PKCS #12 keystores, you might need a password for any interaction with the keystore.

Additional passwords can further protect private keys. This approach is often the same as with the keystore password, but the password can be different. This tactic is useful when a single keystore is shared for

multiple purposes, for example, and when merely having access to the keystore does not guarantee access to all of the data that it contains.

> ⓘ **Note:**
>
> A truststore is another name for a keystore that is intended primarily for use when determining whether to trust a certificate chain that has been presented. Truststores generally contain primarily trusted certificate entries, but that case is not required.

Java runtime environments typically include a default truststore, often `jre/lib/security/cacerts` or `lib/security/cacerts`, that is prepopulated with several widely trusted certification authority certificates. When presented with a certificate that one of these authorities has signed, the default truststore can allow the certificate to be trusted without any additional configuration. When presented with a self-signed certificate, or when presented with a certificate that is signed by an issuer not in the default truststore, such as a private corporate certification authority, a separate truststore is required.

Transport Layer Security (TLS)
TLS describes a mechanism for securely communicating between two parties that might have no prior knowledge of each other.

TLS is the successor to SSL, and the two terms are often used interchangeably, even though such usage might not technically be correct.

> ⓘ **Note:**
>
> SSL remains the more widely recognized term. The abbreviation TLS occasionally generates confusion with the StartTLS extended operation, particularly in LDAP.

TLS provides security in the form of the following main components:

Certificate trust

Is about reassuring a connection-initiating client that it is communicating with the server to which it intended to connect. To ensure that the server shares the same degree of confidence in the identity and legitimacy of the client, it can ask the client to present its own certificate chain. For more information, see *Certificate Trust*.

Cipher selection

Involves choosing the cipher and the key to protect the bulk of the communication. Although a client can use a server certificate's public key to encrypt data before sending it, this approach can lead to the following issues:

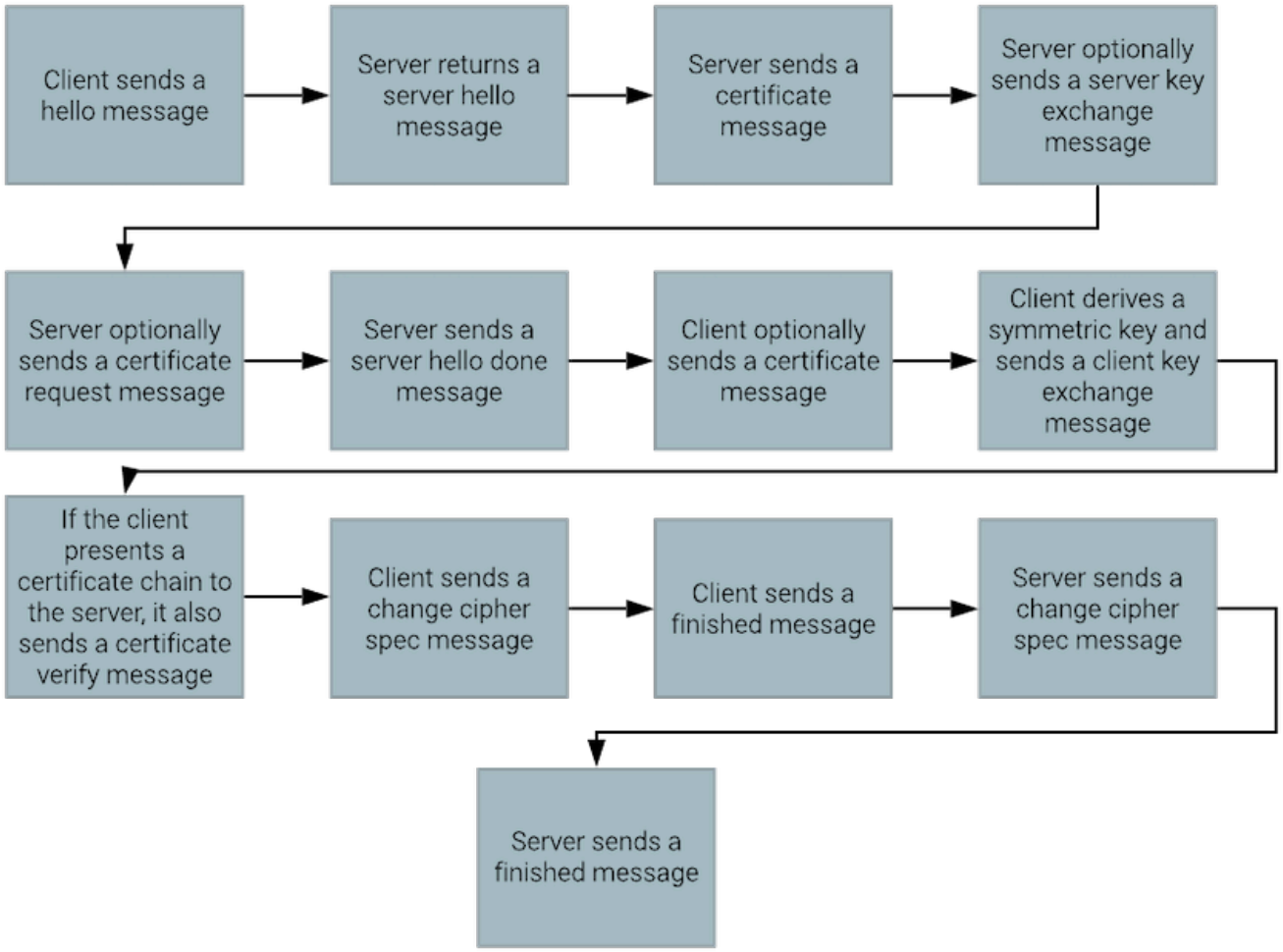
- Unless the client presents its own certificate chain to the server, the server cannot encrypt the data that it sends back to the client.
- Public key encryption is considerably slower than symmetric encryption, in which the same key is used for both encryption and decryption. Public key encryption is also called asymmetric encryption because different keys are used to encrypt and decrypt data.
- If you rely entirely on the security of a private key to ensure the secrecy of a communication, and if the private key becomes compromised, data that has been encrypted with the private key must also be considered compromised.

Rather than relying solely on public key encryption to protect communication between a client and server, the TLS negotiation process allows a client and server to agree on the type of encryption and the secret key to use after completing the negotiation process.

TLS handshakes

The process of negotiating the TLS is referred to as the handshake.

Although the exact process depends on the TLS version that is ultimately chosen, the following steps represent the basic components of a TLS 1.2 handshake:



TLS processing steps

1. The client sends a `hello` message that provides the server with the following information:
   - Highest supported version of the TLS protocol
   - The cipher suites that the client uses
   - Set of extensions with additional information:
     - The address that the client uses to communicate with the server
     - The signature algorithms and elliptic curves that the client supports
     - Whether the client supports secure renegotiation
2. The server returns a `server hello` message that provides the client with the following information:
   - The TLS protocol version that the server uses
   - The cipher suite that the server selects

   The server can also provide its own extensions to the client.

3. The server sends a `certificate` message that provides its certificate chain to the client.
4. The server can optionally send a `server key exchange` message with additional information that the client might need to securely derive the same symmetric encryption key that the server generates.
5. The server can optionally send a `certificate request` message that asks the client to present its own certificate chain to the server.
6. The server sends a `server hello done` message to inform the client that it has completed its `hello` sequence.
7. The client can optionally send a `certificate` message to the server with its own certificate chain.

> ⓘ **Note:**
>
> The client sends a `certificate` message only when the server initially sends a certificate request. If the client receives such a request, it can refuse to, and probably will not, send a certificate chain. The server decides whether to require a client certificate chain. In LDAP, the server commonly asks the client to present a certificate, but continues with TLS negotiation even if the client does not present one. This approach supports authentication methods like SASL EXTERNAL, in which a client uses the certificate chain that it presents during TLS negotiation as proof of its identity.

8. The client derives a symmetric key to use for the remainder of the encrypted processing, and sends a `client key exchange` message to the server. The `client key exchange` message includes the information that the server needs to generate the same key. Only the client and server know the value of the key, even if another entity can observe the communication that passes between the client and the server.
9. If the client presents a certificate chain to the server, it also sends a `certificate verify` message to prove that the private key for the certificate is included at the head of the chain.
10. The client sends a `change cipher spec` message to the server, which informs the server that the client will use the agreed-upon symmetric key to encrypt everything else that it sends to the server.
11. The client sends a `finished` message to the server to indicate that it has completed its portion of the handshake.
12. The server sends a `change cipher spec` message to the client, which informs the client that the server will use the agreed-upon symmetric key to encrypt everything else that it sends to the client.
13. The server sends a `finished` message to the client to indicate that it has completed its portion of the handshake.

TLS 1.3 uses a different handshake sequence that can require only a single round-trip to exchange the necessary information between the client and the server. TLS 1.2 requires two round-trips. To accomplish this task, TLS 1.3 tries to guess the type of key agreement that the server wants to use, and sends the relevant information to the server up front instead of waiting to hear from the server.

Because an extra round of communication between the client and server is eliminated, the server finishes its portion of the negotiation before the client. The server must assume that the client trusts its certificate chain. Because the server might log a successful negotiation only to discover late, through a TLS alert, that the client rejected the certificate, this approach might complicate certain types of troubleshooting.

Key agreement

Key agreement processing provides a critical component of TLS negotiation.

It allows the client and server to select the symmetric key that encrypts the remainder of the communication, but does not reveal the key to anyone who can access the communication. Although several key agreement algorithms are available, the following types are the most common:

RSA key exchange

The client generates random data, uses the server's public key to encrypt it, and provides it to the server, which uses its private key to decrypt it. The client and server alike derive the encryption key from the randomly generated data.

Diffie-Hellman (DH) key exchange

The client and server agree publicly on a pair of mathematically linked numbers, and each participant chooses its own secret value. Through a special computation, they generate a key that can be discovered only by someone who knows one of the secret values. Although several variants of the Diffie-Hellman algorithm can be used in key exchange, we recommend the ECHDE and DHE versions because they use ephemeral keys with no relation to the server's certificate. Of those two versions, ECDHE is faster and uses smaller keys.

When possible, use ECHDE over DHE, and either of those options over RSA. The DH algorithms provide a substantial benefit over RSA in the form of forward secrecy. Because RSA key exchange uses the server certificate's public key to encrypt data, the encryption can be broken if the certificate's private key is compromised. This warning applies to previously captured data as well as to communication on new TLS connections. The use of ephemeral keys in ECDHE and DHE ensures that, even if the certificate's private key is compromised, the encrypted communication remains indecipherable to anyone but the client and server, although anyone with the private key can still impersonate the legitimate server.

LDAP StartTLS extended operation

In most scenarios, a client that uses TLS establishes a connection to a port that is dedicated to its use, like 636 (LDAPS) or 443 (HTTPS).

The client begins the TLS-negotiation process by sending a `client hello` message over the connection. In some scenarios, the client establishes a non-secure connection and later converts it to a secure one. In LDAP, this task is accomplished by using the **`StartTLS`** extended operation.

The **`StartTLS`** extended operation provides the following advantages over a dedicated LDAPS connection:

- To enable secure as well as insecure communication, only one port needs to be opened through a firewall.
- A client can use opportunistic encryption, in which the client performs the following steps:

  1. Queries the root DSE to determine whether the server supports StartTLS.
  2. Secures the connection, if possible.

  Opportunistic encryption is useful in scenarios like following referrals because LDAP URLs do not officially support LDAPS as a scheme.

To ensure that a communication is always secure, use LDAPS instead of establishing an insecure connection that you secure later with the **`StartTLS`** extended operation. If you enable support for unencrypted LDAP communication, as **`StartTLS`** requires, a client might send a password-containing bind request or other sensitive data over an unencrypted connection. A server can be configured to reject unencrypted communication, but it cannot prevent a client from sending an unencrypted request.

> ⓘ **Note:**
>
> Although you can use **`StartTLS`** to temporarily secure a connection before falling back on an unencrypted LDAP communication, PingDirectoryProxy Server does not support this strategy.

About the manage-certificates tool

PingDirectory Server offers a **`manage-certificates`** tool that enables interaction with Java KeyStore (JKS) and PKCS #12 key stores.

Although it behaves similarly to the **`keytool`** utility that accompanies most Java distributions, **`manage-certificates`** is easier to use, provides improved usage information, and offers additional functionality.

Available subcommands

The **`manage-certificates`** tool uses the following subcommands to indicate which function to invoke:

Subcommand	Function
list-certificates	Lists the certificates in a keystore.
import-certificate	Imports a certificate into a trusted certificate entry or imports a certificate chain and private key into a private key entry.
export-certificate	Exports a certificate from a keystore.
export-private-key	Exports a private key from a keystore.
generate-self-signed-certificate	Generates a self-signed certificate.
generate-certificate-signing-request	Generates a certificate-signing request that can be provided to a certification authority.
sign-certificate-signing-request	Signs a certificate-signing request with a specified issuer certificate.
check-certificate-usability	Checks a specified certificate in a keystore to verify whether it is suitable for use as a listener certificate.
trust-server-certificate	Initiates the TLS-negotiation process with a specified server to obtain its certificate chain so that a truststore can be updated with the necessary information to trust the chain.
display-certificate-file	Displays the contents of a file that contains one or more PEM-encoded or DER-encoded X.509 certificates.
display-certificate-signing-request-file	Displays the contents of a file that contains a PEM-encoded or DER-encoded PKCS #10 certificate-signing request (CSR).
change-certificate-alias	Changes the alias for an entry in a keystore.
change-keystore-password	Changes the password for a keystore.
change-private-key-password	Changes the password that protects the private key for a specified entry in a keystore.

Common arguments

Most of the **manage-certificates** subcommands require access to a Java KeyStore (JKS) or PKCS #12 keystore. In such instances, use the --keystore argument to specify the path to the keystore.

If the keystore already exists, the tool detects automatically whether it is a JKS or PKCS #12 keystore. If the operation creates a new keystore, you can specify the type explicitly by using the --keystore-type argument, followed by a value of JKS or PKCS12. If you do not specify the keystore type, a default value of JKS is used.

Some situations require you to provide the password that is needed to access the keystore. For a JKS keystore, you might need to provide a keystore password only for operations that involve creating a keystore or accessing a private key. However, you will likely need to provide the password for all operations that involve a PKCS #12 keystore.

To provide a keystore password, use one of the following arguments:

- --keystore-password, followed by the clear-text password for the keystore.
- --keystore-password-file, followed by the path to a file that contains the password for the keystore. The file might contain the password in the clear, or it might be encrypted with a definition from the server's encryption-settings database.
- --prompt-for-keystore-password. If this argument is provided, the tool prompts you interactively to provide the password.

If a private key is protected with a different password than the keystore itself, specify one of the following arguments to provide the private key password:

- --private-key-password, followed by the plaintext password.
- --private-key-password-file, followed by the path to a file that contains the clear-text or encrypted password.
- --prompt-for-private-key-password, which causes the tool to prompt interactively for the password.

Several operations require you to specify the keystore entry to target. In such scenarios, provide the --alias argument, followed by the name of the alias for that entry.

Listing the certificates in a keystore
List the certificates available in a keystore.

Steps

- To list the certificates in a keystore, use the **`list-certificates`** subcommand.

  This subcommand requires you to specify the path to the keystore file, and possibly the password that is needed to access the keystore. The following options are also available:

Option	Description
--alias {alias}	Specifies the alias of the certificate to display. If this value is not provided, all certificates are displayed. To list more than one specific certificate, specify this value multiple times.
--display-pem-certificate	Includes a PEM-encoded representation of each certificate as part of the output.
--verbose	Includes details about each certificate.

  The following command demonstrates the basic listing of a keystore that contains a single certificate chain.

```
$ bin/manage-certificates list-certificates \
     --keystore config/keystore \
     --keystore-password-file config/keystore.pin

Alias:  server-cert (Certificate 1 of 2 in a chain)
Subject DN:  CN=ds1.example.com,O=Example Corp,C=US
Issuer DN:  CN=Example Certification Authority,O=Example Corp,C=US
Validity Start Time:  Saturday, November 9, 2019 at 11:26:09 AM CST
                     (8 minutes, 15 seconds ago)
Validity End Time:  Sunday, November 8, 2020 at 11:26:09 AM CST
(364 days, 23 hours, 51 minutes, 44 seconds from now)
Validity State:  The certificate is currently within the validity window.
Signature Algorithm:  SHA-256 with ECDSA
Public Key Algorithm:  EC (secP256r1)
SHA-1 Fingerprint: 42:f8:85:97:bf:88:bc:74:4b:5b:ce:0c:54:43:9b:44:6b:
                   81:23:a3
SHA-256 Fingerprint: 4f:be:47:ed:36:68:13:38:ba:e8:c0:c5:6c:85:51:97:
```

```
                     8b:40:1b:76:10:c0:be:80:15:62:06:96:c5:71:30:df
Private Key Available:  Yes
The certificate has a valid signature.

Alias:  server-cert (Certificate 2 of 2 in a chain)
Subject DN:  CN=Example Certification Authority,O=Example Corp,C=US
Issuer DN:  CN=Example Certification Authority,O=Example Corp,C=US
Validity Start Time: Saturday, November 9, 2019 at 11:26:08 AM CST
                    (8 minutes, 16 seconds ago)
Validity End Time: Friday, November 4, 2039 at 12:26:08 PM CDT
                   (7299 days, 23 hours, 51 minutes, 43 seconds from now)
Validity State:  The certificate is currently within the validity window.
Signature Algorithm:  SHA-256 with ECDSA
Public Key Algorithm:  EC (secP256r1)
SHA-1 Fingerprint: b8:d0:16:9b:5d:f2:e7:a1:80:79:95:a2:64:b5:aa:ad:80:
                   23:64:16
SHA-256 Fingerprint: cf:98:2a:66:35:6e:6d:f9:5d:25:c6:68:68:04:5a:a8:
                     88:43:ca:b5:c8:e5:c9:95:09:e9:fc:ab:b9:41:ec:71
The certificate has a valid signature.
```

The following sample represents the verbose version of the previous command.

```
$ bin/manage-certificates list-certificates \
     --keystore config/keystore \
     --keystore-password-file config/keystore.pin \
     --verbose

Alias:  server-cert (Certificate 1 of 2 in a chain)
X.509 Certificate Version:  v3
Subject DN:  CN=ds1.example.com,O=Example Corp,C=US
Issuer DN:  CN=Example Certification Authority,O=Example Corp,C=US
Serial Number:  7b:2d:91:6a:ff:51:4f:7a:19:16:26:4f:ce:cb:cb:31
Validity Start Time:  Saturday, November 9, 2019 at 11:26:09 AM CST
(9 minutes, 48 seconds ago)
Validity End Time: Sunday, November 8, 2020 at 11:26:09 AM CST
                   (364 days, 23 hours, 50 minutes, 11 seconds from now)
Validity State:  The certificate is currently within the validity window.
Signature Algorithm:  SHA-256 with ECDSA
Signature Value:
     30:46:02:21:00:cb:d5:5e:45:b2:8a:33:5e:2d:85:23:39:49:d1:3f:8f:dc:
     f8:9e:2f:f3:44:2f:41:0d:69:95:ec:f0:f5:c0:80:02:21:00:ef:8f:32:35:
     3c:88:f4:89:ed:f3:a6:76:
     bb:92:6c:eb:c6:17:ac:61:dc:67:26:f0:ec:67:90:51:28:a1:d0:d5
Public Key Algorithm:  EC (secP256r1)
Elliptic Curve Public Key Is Compressed:  false
Elliptic Curve X-Coordinate:
   -24253153720011259408467676608081666342358203254369897642016197975874105796326
Elliptic Curve Y-Coordinate:
   48722714538591494552787288916186748185323678082126843165293664643134352536146
Certificate Extensions:
     Subject Key Identifier Extension:
          OID:  2.5.29.14
          Is Critical:  false
          Key Identifier:
               21:ad:b9:7a:15:e4:08:13:05:e1:c2:64:0c:86:aa:9b:f0:4c:fb:a0
     Authority Key Identifier Extension:
          OID:  2.5.29.35
          Is Critical:  false
          Key Identifier:
               01:4b:69:99:93:5f:76:51:39:95:61:cc:a9:a8:cb:16:f2:0f:8c:c8
     Subject Alternative Name Extension:
```

```
          OID:  2.5.29.17
          Is Critical:  false
          DNS Name:  ds1.example.com
          DNS Name:  ds.example.com
          DNS Name:  ldap.example.com
          DNS Name:  localhost
          IP Address:  127.0.0.1
          IP Address:  0:0:0:0:0:0:0:1
     Key Usage Extension:
          OID:  2.5.29.15
          Is Critical:  false
          Key Usages:
               Digital Signature
               Key Encipherment
               Key Agreement
     Extended Key Usage Extension:
          OID:  2.5.29.37
          Is Critical:  false
          Key Purpose ID:  TLS Server Authentication
          Key Purpose ID:  TLS Client Authentication
SHA-1 Fingerprint:
   42:f8:85:97:bf:88:bc:74:4b:5b:ce:0c:54:43:9b:44:6b:81:23:a3
SHA-256 Fingerprint:
   4f:be:47:ed:36:68:13:38:ba:e8:c0:c5:6c:85:51:97:8b:40:1b:76:
   10:c0:be:80:15:62:06:96:c5:71:30:df
Private Key Available:  Yes
The certificate has a valid signature.

Alias:  server-cert (Certificate 2 of 2 in a chain)
X.509 Certificate Version:  v3
Subject DN:  CN=Example Certification Authority,O=Example Corp,C=US
Issuer DN:  CN=Example Certification Authority,O=Example Corp,C=US
Serial Number:  43:b7:bb:0c:82:58:42:d8:06:fc:2a:f6:04:e8:2e:8c
Validity Start Time: Saturday, November 9, 2019 at 11:26:08 AM CST
                     (9 minutes, 49 seconds ago)
Validity End Time: Friday, November 4, 2039 at 12:26:08 PM CDT
                   (7299 days, 23 hours, 50 minutes, 10 seconds from now)
Validity State:  The certificate is currently within the validity window.
Signature Algorithm:  SHA-256 with ECDSA
Signature Value:
     30:45:02:21:00:b9:87:50:5d:b7:6a:19:82:99:9b:aa:f1:5d:25:a1:90:3c:
     17:9d:7f:f5:7f:8d:06:b4:57:41:9e:15:c6:5a:af:02:20:0c:00:5e:17:bf:
     ca:bf:0b:ff:db:9f:dc:55:ad:35:eb:df:f6:37:4e:23:83:36:88:d2:cc:
     7d:9e:23:da:78:28
Public Key Algorithm:  EC (secP256r1)
Elliptic Curve Public Key Is Compressed:  false
Elliptic Curve X-Coordinate:

 -20753103001920939059800335367415761738764700353772539765405069978726324039 64
Elliptic Curve Y-Coordinate:

 67079356503908427292378918440889412002659485731683570737365127953554508553 73
Certificate Extensions:
     Subject Key Identifier Extension:
          OID:  2.5.29.14
          Is Critical:  false
          Key Identifier:
               01:4b:69:99:93:5f:76:51:39:95:61:cc:a9:a8:cb:16:f2:0f:8c:c8
     Basic Constraints Extension:
          OID:  2.5.29.19
          Is Critical:  false
          Is CA:  true
          Path Length Constraint:  0
     Key Usage Extension:
```

```
          OID:  2.5.29.15
          Is Critical:  false
          Key Usages:
               Key Cert Sign
               CRL Sign
SHA-1 Fingerprint:
 b8:d0:16:9b:5d:f2:e7:a1:80:79:95:a2:64:b5:aa:ad:80:23:64:16
SHA-256 Fingerprint:

 cf:98:2a:66:35:6e:6d:f9:5d:25:c6:68:68:04:5a:a8:88:43:ca:b5:c8:e5:c9:95:09:
   e9:fc:ab:b9:41:ec:71
The certificate has a valid signature.
```

Generating self-signed certificates

The process of creating a self-signed certificate is straightforward because a self-signed certificate claims itself as its own issuer.

Although self-signed certificates are convenient for testing environments, clients do not trust them by default. Consequently, you should not use them as listener certificates in production environments.

The **`manage-certificates`** tool offers a **`generate-self-signed-certificate`** subcommand that can create a self-signed certificate. In addition to the arguments that provide information about the keystore, certificate alias, and optional private key password, the following arguments are available.

Argument	Description
`--subject-dn {subject}`	Subject DN for the certificate to create. This value is required.
`--days-valid {number}`	Number of days that the certificate remains valid. Defaults to 365 if no value is specified.
`--validity-start-time {timestamp}`	Indicates the time at which the certificate begins its validity window. This value is assumed to reflect the local time zone, and must be expressed in the form YYYYMMDDhhmmss, where a value of 20190102030405 indicates January 2, 2019, at 3:04:05 AM. Defaults to the current time if no value is specified.
`--key-algorithm {name}`	Name of the algorithm to use when generating the key pair. For a listener certificate, this value is typically RSA or EC. Defaults to RSA if no value is specified. **Note:** This argument cannot be used in conjunction with the `--replace-existing-certificate` argument.

Argument	Description
`--key-size-bits {number}`	Length of the key, in bits, to generate. If the `--key-algorithm` argument is given, then `--key-size-bits {number}` must also be specified. Conversely, if the `--replace-existing-certificate` argument is given, then `--key-size-bits {number}` must not be specified. Typical key sizes are: ▪ RSA key – 2048 or 4096 bits If a default RSA key is used but this argument is not provided, a default key size of 2048 bits is used. ▪ Elliptic curve key – 256 or 384 bits
`--signature-algorithm {name}`	Name of the algorithm to use to sign the certificate. If the `--key-algorithm` argument is used to specify an algorithm other than RSA, then `--signature-algorithm {name}` must also be specified. If the `--replace-existing-certificate` argument is used, then `--signature-algorithm {name}` must not be specified. Typical signature algorithms include SHA256withRSA for certificates with RSA keys, and SHA256withECDSA for certificates with elliptic curve keys. If a default key algorithm is used but the `--signature-algorithm {name}` argument is not provided, a default value of SHA256withRSA is used.
`--replace-existing-certificate`	Uses the new certificate to replace an existing certificate in the key store (within the same alias), and reuses the key for that certificate.
`--inherit-extensions`	Indicates that, when replacing an existing certificate, the new certificate contains the same set of extensions as the existing certificate. If the `--replace-existing-certificate` argument is provided, but the `--inherit-extensions` argument is omitted, the new certificate contains only arguments that are provided explicitly.
`--subject-alternative-name-dns {name}`	Indicates that the certificate is expected to have a subject alternative name extension with the provided DNS name. The given name must be fully qualified, although it can contain an asterisk (*) as a wildcard in the leftmost component. To include multiple DNS names in the subject alternative name extension, specify the `--subject-alternative-name-dns {name}` argument multiple times.

Argument	Description
`--subject-alternative-name-ip-address {address}`	Indicates that the certificate is expected to have a subject alternative name extension with the provided IP address. The given address must be a valid IPv4 or IPv6 address. No wildcards are allowed. To include multiple IP addresses in the subject alternative name extension, specify the `--subject-alternative-name-ip-address {address}` argument multiple times.
`--subject-alternative-name-email-address {address}`	Indicates that the certificate is expected to have a subject alternative name extension with the provided email address. To include multiple email addresses in the subject alternative name extension, specify the `--subject-alternative-name-email-address {address}` argument multiple times.
`--subject-alternative-name-uri {uri}`	Indicates that the certificate is expected to have a subject alternative name extension with the provided URI. To include multiple URIs in the subject alternative name extension, specify the `--subject-alternative-name-uri {uri}` argument multiple times.
`--subject-alternative-name-oid {oid}`	Indicates that the certificate is expected to have a subject alternative name extension with the provided object identifier (OID). The given value must be a valid OID. To include multiple OIDs in the subject alternative name extension, specify the `--subject-alternative-name-oid {oid}` argument multiple times.
`--basic-constraints-is-ca {value}`	Indicates that the certificate is expected to have a basic constraints extension, with a specified value of true or false, for the flag indicating whether to consider the certificate a certification authority that can be used to sign other certificates. ▪ For root and intermediate certificate authority (CA) certificates, the `--basic-constraints-is-ca {value}` argument must be present with a value of true. ▪ For end-entity certificates, the `--basic-constraints-is-ca {value}` argument can optionally be present with a value of false. ▪ For a self-signed certificate, specify the `--basic-constraints-is-ca {value}` argument with a value of false to indicate that the certificate is not considered a CA certificate.

Argument	Description
`--basic-constraints-maximum-path-length {number}`	Indicates that the basic constraints extension is expected to include a path length constraint element with the specified value. Use this argument only if `--basic-constraints-is-ca` is provided with a value of true. A path length constraint value of 0 indicates that the certificate can be used to issue only end-entity certificates. A path length constraint value of 1 indicates that the certificate can be used to sign end-entity certificates or intermediate CA certificates, the latter of which can be used to sign only end-entity certificates. A value greater than 1 indicates the presence of several intermediate CA certificates between it and the end-entity certificate at the head of the chain.
`--key-usage {value}`	Indicates that the certificate is expected to have a key usage extension with the specified value. The following values are allowed: ▪ digital-signature ▪ non-repudiation ▪ key-encipherment ▪ data-encipherment ▪ key-agreement ▪ key-cert-sign ▪ crl-sign ▪ encipher-only ▪ decipher-only To include multiple key usages, specify the `--key-usage {value}` argument multiple times.
`--extended-key-usage {value}`	Indicates that the certificate is expected to have an extended key usage extension with the specified value. The following values are allowed: ▪ server-auth ▪ client-auth ▪ code-signing ▪ email-protection ▪ time-stamping ▪ ocsp-signing

For example, the following command can be used to generate a self-signed server certificate.

```
bin/manage-certificates generate-self-signed-certificate \
     --keystore config/keystore \
     --keystore-password-file config/keystore.pin \
     --keystore-type JKS \
     --alias server-cert \
```

```
     --subject-dn "CN=ds.example.com,O=Example Corp,C=US" \
     --key-algorithm EC \
     --key-length-bits 256 \
     --signature-algorithm SHA256withECDSA \
     --subject-alternative-name-dns ds.example.com \
     --subject-alternative-name-dns ds1.example.com \
     --subject-alternative-name-dns localhost \
     --subject-alternative-name-ip-address 1.2.3.4 \
     --subject-alternative-name-ip-address 127.0.0.1 \
     --subject-alternative-name-ip-address 0:0:0:0:0:0:0:1 \
     --key-usage digital-signature \
     --key-usage key-encipherment \
     --key-usage key-agreement \
     --extended-key-usage server-auth \
     --extended-key-usage client-auth

Successfully created a new JKS keystore.

Successfully generated the following self-signed certificate:
Subject DN:  CN=ds.example.com,O=Example Corp,C=US
Issuer DN:  CN=ds.example.com,O=Example Corp,C=US
Validity Start Time: Monday, January 27, 2020 at 03:40:13 PM
 CST
                     (0 seconds ago)
Validity End Time: Tuesday, January 26, 2021 at 03:40:13 PM CST
                   (364 days, 23 hours, 59 minutes, 59 seconds
 from now)
Validity State:  The certificate is currently within the
 validity window.
Signature Algorithm:  SHA-256 with ECDSA
Public Key Algorithm:  EC (secP256r1)
SHA-1 Fingerprint:
 4f:41:82:7f:08:e9:d8:05:8c:19:8b:3e:5b:bc:59:98:d3:15:71:3a
SHA-256 Fingerprint:

 76:e6:8e:c5:c8:8d:27:ce:2b:85:b9:8c:9d:49:3c:06:f4:40:f1:d0:70:67:39:24:fc:
   31:bc:f8:51:83:f2:42
```

Generating certificate signing requests

A certificate signing request (CSR) contains all of the information that a certification authority requires to issue a certificate.

RFC 2986 defines the request format, also known as PKCS #10, and includes the following elements:

- Certificate signing request version
- Requested subject distinguished name (DN) for the certificate
- Public key for the requested certificate
- Requested set of extensions for the certificate
- Signature that proves the requester has the private key for the given public key

To create a certificate signing request, use the **`manage-certificates generate-certificate-signing-request`** command, which performs the following steps:

1. Generated a public and private key pair.
2. Stores the key pair in a key store with a given alias.
3. Outputs the certificate signing request to the terminal.
4. Optionally writes the certificate signing request to a file.

Because a certificate signing request contains many of the same elements as a certificate, the command to generate one takes most of the same arguments as for generating a self-signed certificate. The following arguments are unavailable when generating a CSR:

- `--replace-existing-certificate`
- `--days-valid {number}`
- `--validity-start-time {timestamp}`

The following arguments are available when generating a certificate signing request but not when generating a self-signed certificate:

`--output-file {path}`

Path to a file to which the certificate signing request is written. If this value is not provided, the request is written only to the terminal in PEM form.

`--output-format {value}`

Format to use when writing the certificate signing request. This value can be PEM or DER, but the DER format is used only in conjunction with the `--output-file` argument. Defaults to PEM if the `--output-format {value}` argument is not provided.

`--use-existing-key-pair`

Indicates that the CSR uses a key pair that already exists in the key store with the given alias, rather than generating a new key pair, in which case the specified alias must not already be in use in the key store.

The following example command creates a CSR.

```
bin/manage-certificates generate-certificate-signing-request \
     --output-file ds1-cert.csr \
     --output-format PEM \
     --keystore config/keystore \
     --keystore-password-file config/keystore.pin \
     --keystore-type JKS \
     --alias server-cert \
     --subject-dn "CN=ds.example.com,O=Example Corp,C=US" \
     --key-algorithm EC \
     --key-length-bits 256 \
     --signature-algorithm SHA256withECDSA \
     --subject-alternative-name-dns ds.example.com \
     --subject-alternative-name-dns ds1.example.com \
     --subject-alternative-name-dns localhost \
     --subject-alternative-name-ip-address 1.2.3.4 \
     --subject-alternative-name-ip-address 127.0.0.1 \
     --subject-alternative-name-ip-address 0:0:0:0:0:0:0:1 \
     --key-usage digital-signature \
     --key-usage key-encipherment \
     --key-usage key-agreement \
     --extended-key-usage server-auth \
     --extended-key-usage client-auth

Successfully created a new JKS keystore.

Successfully generated the key pair to use for the certificate signing
request.

Successfully wrote the certificate signing request to file
'/ds/build/package/PingDirectory/ds1-cert.csr'.
```

If the contents of the resulting CSR file are made available to a certification authority to be signed, the resulting signed certificate can be imported into the key store.

To print the contents of a certificate signing request file, use the **`display-certificate-signing-request-file`** subcommand, which supports the following arguments:

--certificate-signing-request-file {path}

Path to the file that contains the certificate signing request to display.

--verbose

Indicates that the command is expected to display verbose information about the request, rather than a basic information set.

The following example demonstrates the basic output from the command.

```
$ bin/manage-certificates display-certificate-signing-request-file \
    --certificate-signing-request-file ds1-cert.csr

    PKCS #10 Certificate Signing Request Version:  v1
    Subject DN:  CN=ds.example.com,O=Example Corp,C=US
    Signature Algorithm:  SHA-256 with ECDSA
    Public Key Algorithm:  EC (secP256r1)
```

The following example demonstrates the verbose output.

```
$ bin/manage-certificates display-certificate-signing-request-file \
     --certificate-signing-request-file ds1-cert.csr \
     --verbose

PKCS #10 Certificate Signing Request Version:  v1
Subject DN:  CN=ds.example.com,O=Example Corp,C=US
Signature Algorithm:  SHA-256 with ECDSA
Signature Value:

 30:45:02:20:46:31:be:9e:6d:2f:0e:e3:d0:80:5c:88:ef:da:86:07:fd:15:b7:62:
     83:45:39:0a:c9:f2:f9:17:eb:08:94:ff:02:21:00:c8:bd:df:57:fa:ea:8c:04:
     df:c5:27:76:e5:b3:3b:4f:df:ec:d3:e4:09:5b:c0:6c:7b:86:39:ec:d0:0e:c1:64
Public Key Algorithm:  EC (secP256r1)
Elliptic Curve Public Key Is Compressed:  false
Elliptic Curve X-Coordinate:
   208628537904757963197889471667098239762296638799662436502070112279302 4
   3221133
Elliptic Curve Y-Coordinate:
   479697739226644990505743464941788269420922508654777168408919906254139
   60212095
Certificate Extensions:
     Subject Key Identifier Extension:
          OID:  2.5.29.14
          Is Critical:  false
          Key Identifier:
               f2:de:fd:bf:d3:2f:96:ef:01:70:2d:0e:85:f5:fb:17:d5:a0:9e:67
     Subject Alternative Name Extension:
          OID:  2.5.29.17
          Is Critical:  false
          DNS Name:  ds.example.com
          DNS Name:  ds1.example.com
          DNS Name:  localhost
          IP Address:  1.2.3.4
          IP Address:  127.0.0.1
          IP Address:  0:0:0:0:0:0:0:1
     Key Usage Extension:
          OID:  2.5.29.15
          Is Critical:  false
          Key Usages:
               Digital Signature
               Key Encipherment
```

```
               Key Agreement
     Extended Key Usage Extension:
          OID:  2.5.29.37
          Is Critical:  false
          Key Purpose ID:  TLS Server Authentication
          Key Purpose ID:  TLS Client Authentication
```

Importing signed and trusted certificates

Use the **`manage-certificates import-certificate`** command to import certificates into a keystore.

This command is used to accomplish the following tasks:

- Import a certificate that a certification authority has signed into the keystore in which the key pair was generated. In this scenario, the certificate is imported into a private key entry and must be imported as a certificate chain rather than an end-entity certificate.
- Import a trusted issuer certificate into a trust store. In this scenario, the certificate is imported into a trusted certificate entry as a single certificate instead of as a chain.
- Import a certificate chain, along with the private key for the end-entity certificate. This approach imports certificates that were generated through another library, like OpenSSL.

In addition to the arguments that provide information about the key store and the alias into which the certificate or certificate chain is imported, the **`manage-certificates import-certificate`** command accepts the following arguments:

`--certificate-file {path}`

Path to the file that contains the certificate to import. The certificate can be in PEM or DER format and can be a single certificate or a certificate chain. If the certificates in the chain reside in separate files, specify the --certificate-file {path} argument multiple times when you import a certificate chain.

`--private-key-file {path}`

Path to the file containing the private key that corresponds to the certificate at the head of the imported chain. The private key can be in PEM or DER format.

`--no-prompt`

Indicates that the certificate is to be imported without prompting for confirmation. By default, a summary of the certificate is displayed, and you must confirm that you want to import it.

The following example command imports a signed certificate into the key store that generates the certificate signing request.

```
$ bin/manage-certificates import-certificate \
     --keystore config/keystore \
     --keystore-password-file config/keystore.pin \
     --alias server-cert \
     --certificate-file ds1-cert.pem \
     --certificate-file ca-cert.pem

The following certificate chain will be imported into the keystore into
 alias
'server-cert', preserving the existing private key associated with that
 alias:

Subject DN:  CN=ds.example.com,O=Example Corp,C=US
Issuer DN:  CN=Example Root CA,O=Example Corp,C=US
Validity Start Time: Sunday, November 10, 2019 at 09:09:23 PM CST
                     (4 minutes, 16 seconds ago)
Validity End Time: Monday, November 9, 2020 at 09:09:23 PM CST
                   (364 days, 23 hours, 55 minutes, 43 seconds from now)
Validity State:  The certificate is currently within the validity window.
```

```
Signature Algorithm:  SHA-256 with ECDSA
Public Key Algorithm:  EC (secP256r1)
SHA-1 Fingerprint:
 02:51:25:43:3e:68:f5:71:36:e3:5d:df:74:de:f6:a1:5a:db:0f:eb
SHA-256 Fingerprint: 1d:b5:eb:3c:f5:ff:bf:79:a2:a5:86:b8:e4:33:76:4d:d7:
                     50:dc:a4:34:95:37:be:89:45:86:1f:5d:79:c3:93

Subject DN:  CN=Example Root CA,O=Example Corp,C=US
Issuer DN:  CN=Example Root CA,O=Example Corp,C=US
Validity Start Time: Sunday, November 10, 2019 at 09:00:07 PM CST
                     (13 minutes, 32 seconds ago)
Validity End Time: Saturday, November 5, 2039 at 10:00:07 PM CDT
                   (7299 days, 23 hours, 46 minutes, 27 seconds from now)
Validity State:  The certificate is currently within the validity window.
Signature Algorithm:  SHA-256 with ECDSA
Public Key Algorithm:  EC (secP384r1)
SHA-1 Fingerprint:
 0e:5c:21:c9:a5:36:0a:24:eb:aa:55:b6:a5:94:0e:e0:56:03:22:e6
SHA-256 Fingerprint:
 77:cf:66:d7:3c:8a:fd:67:2d:b7:36:fd:60:1d:ca:eb:1b:03:b1:
                     12:7b:10:1f:26:05:b7:b9:0d:02:e0:38:3e

Do you want to import this certificate chain into the keystore? yes

Successfully imported the certificate chain.
```

If you do not provide the `--no-prompt` argument, the **`manage-certificates import-certificate`** tool still displays information about the certificates to import. To view additional information about a certificate before you import it, use the **`display-certificate-file`** subcommand, which supports the following arguments:

`--certificate-file {path}`

Path to the file that contains the certificate to view.

`--verbose`

Displays verbose information about the certificate.

The output of the **`display-certificate-file`** subcommand has the same format and content as the **`list-certificates`** subcommand.

Exporting certificates

Use the **`export-certificate`** subcommand to export a single certificate or a certificate chain from a key store to a file in PEM or DER format.

The **`export-certificate`** subcommand supports the normal arguments about the key store and certificate alias, in addition to the following arguments:

`--output-file {path}`

Path to the file to which exported certificates are written. If this value is not provided, the certificates are written to standard output rather than a file.

`--output-format {format}`

Format in which exported certificates are written. The value can be PEM or DER, but the DER format can be used only if the output is written to a file. Defaults to PEM if no value is specified.

`--export-certificate-chain`

Indicates that a certificate chain, rather than the end-entity certificate only, is to be exported.

`--separate-file-per-certificate`

Indicates the use of separate output files for each exported certificate, rather than placing all of the certificates in a single file. If this argument is provided and multiple certificates are to be exported, then .1 is appended to the path for the indicated output file for the first certificate in the chain, .2 is appended for the second certificate, and so on.

The following example exports a certificate chain.

```
$ bin/manage-certificates export-certificate \
     --keystore config/keystore \
     --keystore-password-file config/keystore.pin \
     --alias server-cert \
     --output-file server-cert.pem \
     --output-format PEM \
     --export-certificate-chain \
     --separate-file-per-certificate

Successfully exported the following certificate to '/ds/server-cert.pem.1':
Subject DN:  CN=ds.example.com,O=Example Corp,C=US
Issuer DN:  CN=Example Root CA,O=Example Corp,C=US
Validity Start Time: Sunday, November 10, 2019 at 09:09:23 PM CST
                     (3 hours, 26 minutes, 23 seconds ago)
Validity End Time: Monday, November 9, 2020 at 09:09:23 PM CST
                           (364 days, 20 hours, 33 minutes, 36 seconds from
 now)
Validity State:  The certificate is currently within the validity window.
Signature Algorithm:  SHA-256 with ECDSA
Public Key Algorithm:  EC (secP256r1)
SHA-1 Fingerprint:
 02:51:25:43:3e:68:f5:71:36:e3:5d:df:74:de:f6:a1:5a:db:0f:eb
SHA-256 Fingerprint:
1d:b5:eb:3c:f5:ff:bf:79:a2:a5:86:b8:e4:33:76:4d:d7:50:dc:a4:34:95:37:be:89:45:
86:1f:5d:79:c3:93

Successfully exported the following certificate to '/ds/server-cert.pem.2':
Subject DN:  CN=Example Root CA,O=Example Corp,C=US
Issuer DN:  CN=Example Root CA,O=Example Corp,C=US
Validity Start Time: Sunday, November 10, 2019 at 09:00:07 PM CST
                     (3 hours, 35 minutes, 39 seconds ago)
Validity End Time: Saturday, November 5, 2039 at 10:00:07 PM CDT
                   (7299 days, 20 hours, 24 minutes, 20 seconds from now)
Validity State:  The certificate is currently within the validity window.
Signature Algorithm:  SHA-256 with ECDSA
Public Key Algorithm:  EC (secP384r1)
SHA-1 Fingerprint:
 0e:5c:21:c9:a5:36:0a:24:eb:aa:55:b6:a5:94:0e:e0:56:03:22:e6
SHA-256 Fingerprint:
   77:cf:66:d7:3c:8a:fd:67:2d:b7:36:fd:60:1d:ca:eb:1b:03:b1:12:7b:10:1f:26:
   05:b7:b9:0d:02:e0:38:3e
```

The **`export-certificate`** subcommand exports only the public portion of a certificate. Its private key is not included. To export the private key, use the **`export-private-key`** subcommand, which supports the following arguments, in addition to the usual key store and alias arguments:

`--output-file {path}`

Path to the file to which the exported private key is written. If this value is not provided, the key is written to standard output rather than a file.

`--output-format {format}`

Format in which the exported private key is written. The value can be PEM or DER, but the DER format is used only if the output is written to a file. Defaults to PEM if no value is specified.

The following code provides an example of the **`export-private-key`** subcommand .

```
$ bin/manage-certificates export-private-key \
     --keystore config/keystore \
     --keystore-password-file config/keystore.pin \
     --alias server-cert \
     --output-file server-cert-key.pem \
     --output-format PEM

Successfully exported the private key.
```

Using manage-certificates as a simple certification authority

If your PingDirectoryProxy Server instances need to support an arbitrary or unknown set of clients, configure them with certificates from a trusted issuer, such as a commercial authority or the free Let's Encrypt service.

About this task

If you control every client that accesses the servers, you might want to create your own internal certification authority so that you have a common issuer for all servers. In such a scenario, the clients need to trust only the certificates that the issuer signs. Commercial and open-source software packages provide full-featured certification authority functionality, but you can use the **`manage-certificates`** tool to create a certificate authority (CA) certificate that you can use to sign certificate-signing requests.

Steps

1. Create a CA certificate.

   A CA certificate is a self-signed certificate that possesses the following extensions:

   - A key usage extension that includes at least the keyCertSign usage
   - A basic constraints extension that identifies the certificate as a CA certificate

   If you do not plan to use an intermediate CA certificate, the basic constraints extension must have a path length constraint of 0. If you plan to use an intermediate CA certificate, the path length constraint must be 1. Because certificates that the CA certificate signs are valid only for as long as all certificates in the chain remain valid, we recommend that you specify a long lifespan for the CA certificate.

   The following example creates a new root CA certificate.

   ```
   $ bin/manage-certificates generate-self-signed-certificate \
        --keystore /ca/root-ca-keystore \
        --keystore-password-file /ca/root-ca-keystore.pin \
        --keystore-type JKS \
        --alias root-ca-cert \
        --subject-dn "CN=Example Root CA,O=Example Corp,C=US" \
        --days-valid 7300 \
        --key-algorithm RSA \
        --key-size-bits 4096 \
        --signature-algorithm SHA256withRSA \
        --basic-constraints-is-ca true \
        --basic-constraints-maximum-path-length 1 \
        --key-usage key-cert-sign \
        --key-usage crl-sign

   Successfully created a new JKS keystore.

   Successfully generated the following self-signed certificate:
   Subject DN:  CN=Example Root CA,O=Example Corp,C=US
   Issuer DN:  CN=Example Root CA,O=Example Corp,C=US
   Validity Start Time: Monday, January 27, 2020 at 03:47:29 PM CST (0
    seconds ago)
   ```

```
Validity End Time: Sunday, January 22, 2040 at 03:47:29 PM CST
                   (7299 days, 23 hours, 59 minutes, 59 seconds from now)
Validity State:  The certificate is currently within the validity window.
Signature Algorithm: SHA-256 with RSA
Public Key Algorithm: RSA (4096-bit)
SHA-1 Fingerprint:
 bc:8e:5b:30:52:ec:03:63:b4:9a:aa:1a:45:a0:fc:84:49:dd:e8:64
SHA-256 Fingerprint:

 d5:47:06:cd:a2:95:42:61:1f:c7:aa:04:16:1e:c1:70:41:c4:44:48:bf:74:20:5f:1c:
   61:e2:aa:40:08:3a:ff
```

2. Export the public portion of the root CA certificate for future reference.

   When you import a signed certificate, you can import the public portion of the root CA certificate as a standalone certificate into trust stores as well as into part of a certificate chain.

3. Create a new certificate signing request to create an intermediate CA certificate,

   The certificate signing request uses essentially the same settings as the root CA. If you anticipate only a single intermediate CA, its basic constraints extension must have a path length constraint of 0, rather than 1, to indicate that it is used only to sign end-entity certificates and that it cannot create subordinate CA certificates by itself.

   The following example command creates a certificate signing request.

```
$ bin/manage-certificates generate-certificate-signing-request \
     --keystore /ca/intermediate-ca-keystore \
     --keystore-password-file /ca/intermediate-ca-keystore.pin \
     --keystore-type JKS \
     --alias intermediate-ca-cert \
     --subject-dn "CN=Example Intermediate CA,O=Example Corp,C=US" \
     --key-algorithm RSA \
     --key-size-bits 4096 \
     --signature-algorithm SHA256withRSA \
     --basic-constraints-is-ca true \
     --basic-constraints-maximum-path-length 0 \
     --key-usage key-cert-sign \
     --key-usage crl-sign \
     --output-file /ca/intermediate-ca-cert.csr \
     --output-format PEM

Successfully created a new JKS keystore.

Successfully generated the key pair to use for the certificate signing
request.

Successfully wrote the certificate signing request to file
'/ca/intermediate-ca-cert.csr'.
```

4. Use the root CA certificate to sign the certificate signing request for the intermediate CA certificate with the **`sign-certificate-signing-request`** subcommand.

   The **`sign-certificate-signing-request`** subcommand takes most of the same arguments as generating a self-signed certificate. The primary differences between the argument sets are as follows:

   - The key store that contains the certificate uses the provided key store arguments to sign the request. To specify the name of the certificate to use when signing the request, use the `--signing-certificate-alias` argument.
   - To specify the path to the file that contains the certificate signing request file to generate, provide a `--request-input-file` argument.
   - To specify the path to the file to which the signed certificate is written, provide a `--certificate-output-file` argument. If this argument is omitted, the PEM representation of the certificate is written to standard output.
   - To specify the format, PEM or DER, in which the certificate is written to the output file, provide an `--output-format` argument.
   - To specify the subject to use for the signed certificate, use the `--subject-dn` argument. To use the subject DN from the certificate signing request, omit this argument.
   - To specify the name of the signature algorithm, use the `--signature-algorithm` argument.

     > ⓘ **Note:**
     >
     > Because the requester generated the key, you cannot specify the key algorithm or the key length.

   - To indicate that the signed certificate includes every extension that is listed in the certificate signing request, use the `--include-requested-extensions` argument. If this argument is not provided, explicitly specify the set of extensions to include.

   The following example command signs the certificate signing request for an intermediate CA certificate.

   ```
   $ bin/manage-certificates sign-certificate-signing-request \
        --keystore /ca/root-ca-keystore \
        --keystore-password-file /ca/root-ca-keystore.pin \
        --signing-certificate-alias root-ca-cert \
        --days-valid 7300 \
        --include-requested-extensions \
        --request-input-file /ca/intermediate-ca-cert.csr \
        --certificate-output-file /ca/intermediate-ca-cert.pem \
        --output-format PEM

   Read the following certificate signing request:

   PKCS #10 Certificate Signing Request Version:  v1
   Subject DN:  CN=Example Intermediate CA,O=Example Corp,C=US
   Signature Algorithm:  SHA-256 with RSA
   Public Key Algorithm:  RSA (4096-bit)

   Do you really want to sign this request? yes

   Successfully wrote the signed certificate to file
   '/ca/intermediate-ca-cert.pem'.
   ```

5. After you obtain the intermediate CA certificate, create secure, offline backups of the root CA certificate.

6. Remove the root CA certificate, or at least its private key, from all systems.

> ⓘ **Note:**
>
> Make certain that all end-entity certificates are signed with the intermediate CA certificate, and that the process is identical to the previous example. Restore the root CA certificate only if you need to sign another intermediate CA certificate.

Enabling TLS support during setup
Enable TLS support in the server.

To enable TLS support in the server, you should complete one of the following tasks during the setup procedure:

- Provide a key store that contains the certificate to use.
- Make the installer generate a self-signed certificate.

When using the **`setup`** tool in interactive mode, it prompts you for the information that it needs to configure secure communication, as shown in the following example.

```
Do you want to enable the Directory Server services (Available State,
Available or Degraded State, Configuration, Consent, Directory REST API,
Documentation, SCIM2, and Swagger UI) and Administrative Console over
HTTPS?  After setup, you can selectively enable or disable individual
services and applications by configuring the HTTPS Connection Handler
(yes / no) [yes]: yes

On which port should the Directory Server accept connections from HTTPS
clients? [443]: 443

On which port should the Directory Server accept connections from LDAP
clients? [389]: 389

Do you want to enable LDAPS? (yes / no) [yes]: yes
On which port should the Directory Server accept connections from LDAPS
clients? [636]: 636

Do you want to enable StartTLS? (yes / no) [yes]: yes

Certificate server options:

    1)  Generate self-signed certificate (recommended for testing purposes
        only)
    2)  Use an existing certificate located on a Java Keystore (JKS)
    3)  Use an existing certificate located on a PKCS12 keystore
    4)  Use an existing certificate on a PKCS11 token

Enter option [1]: 2

Java Keystore (JKS) path: /ca/ds1-keystore
Keystore PIN: {password}

Truststore options:

    1)  Generate a default JKS truststore
    2)  Use an existing JKS truststore
    3)  Use an existing PKCS12 truststore

Enter option [1]: 2

JKS truststore path: /ca/truststore
```

```
Truststore password (can be blank): {password}
```

When using **setup** in non-interactive mode, use the following arguments to configure TLS support.

Argument	Description
`--ldapsPort {port}`	Server enables support for LDAPS (LDAP over TLS) on the specified TCP port.
`--httpsPort {port}`	Server enables support for HTTPS for SCIM, the Directory REST API, and the web-based administration console on the specified TCP port.
`--enableStartTLS`	LDAP connection handler enables support for the StartTLS extended operation.
`--generateSelfSignedCertificate`	**setup** generates a self-signed certificate that is presented to clients that use LDAPS, HTTPS, and the StartTLS extended operation.
`--useJavaKeyStore {path}`	Server uses the specified Java KeyStore (JKS) key store to obtain the certificate chain that it presents to clients that use LDAPS, HTTPS, and the StartTLS extended operation.
`--usePKCS12KeyStore {path}`	Server uses the specified PKCS #12 key store to obtain the certificate chain that it presents to clients that use LDAPS, HTTPS, and the StartTLS extended operation.
`--usePKCS11KeyStore`	Server uses a PKCS #11 key store, like a hardware security module, to obtain the certificate chain that it presents to clients that use LDAPS, HTTPS, and the StartTLS extended operation. The Java Virtual Machine (JVM) must already be configured to access the appropriate key store through PKCS #11.
`--keyStorePassword {password}`	Password that is needed to interact with the specified JKS, PKCS #12, or PKCS #11 key store. The **setup** tool assumes that the private key password matches the key store password.
`--keyStorePasswordFile {path}`	Path to the file that contains the password needed to interact with the specified JKS, PKCS #12, or PKCS #11 key store.
`--certNickname {alias}`	Alias of the private key entry in the specified key store that contains the certificate chain to present to clients during TLS negotiation. This argument is optional but recommended if the key store contains multiple certificates.
`--useJavaTrustStore {path}`	Server uses the specified JKS trust store to determine whether to trust certificate chains that are presented to it during TLS negotiation.

Argument	Description
`--usePKCS12TrustStore {path}`	Server uses the specified PKCS #12 trust store to determine whether to trust certificate chains that are presented to it during TLS negotiation
`--trustStorePassword {password}`	Password that is needed to interact with the specified JKS or PKCS #11 trust store.
`--trustStorePasswordFile {path}`	Path to the file that contains the password needed to interact with the specified JKS or PKCS #11 trust store.

The following example command sets up PingDirectory Server in non-interactive mode with an existing certificate.

```
$ ./setup \
     --no-prompt \
     --acceptLicense \
     --ldapPort 389 \
     --ldapsPort 636 \
     --httpsPort 443 \
     --enableStartTLS \
     --useJavaKeyStore config/keystore \
     --keyStorePasswordFile config/keystore.pin \
     --certNickname server-cert \
     --useJavaTrustStore config/truststore \
     --trustStorePasswordFile config/truststore.pin \
     --baseDN dc=example,dc=com \
     --rootUserDN "cn=Directory Manager" \
     --rootUserPasswordFile root-pw.txt \
     --maxHeapSize 10g \
     --encryptDataWithPassphraseFromFile encryption-settings-password.txt \
     --instanceName ds1 \
     --location Austin \
     --noPropertiesFile


Ping Identity Directory Server 8.0.0.0

Initializing ..... Done
Configuring Directory Server ..... Done
Configuring Certificates ..... Done
Starting Directory Server ..... Done

Access product documentation from docs/index.html
```

Enabling TLS support after setup

If the server has been set up without support for TLS, enable TLS support later by completing the following tasks.

Steps

1. Obtain a certificate chain.

   For more information about obtaining a certificate chain, see *Certificate chains* on page 322. To prepare a Java KeyStore JKS or PKCS #12 key store with an appropriate certificate chain and private key, use the **`manage-certificates`** tool. We also recommend that you create a trust store that the server can use.

2. Configure the key and trust manager providers.

   For more information, see *Configuring key and trust manager providers* on page 350.
3. Configure connection handlers.

   For more information, see *Configuring connection handlers* on page 350.

Configuring key and trust manager providers
After you have a key store, configure a key manager provider to access it.

The server is preconfigured with key manager providers, `JKS` and `PKCS12`, that you can use with JKS or PKCS #12 key stores, respectively. You can update the appropriate key manager provider in most cases to reference the key store that you plan to use. The following code provides an example.

```
dsconfig set-key-manager-provider-prop \
     --provider-name JKS \
     --set enabled:true \
     --set key-store-file:config/keystore \
     --set key-store-pin-file:config/keystore.pin
```

A similar change configures a trust manager provider to reference the appropriate trust store. The following code provides an example.

```
dsconfig set-trust-manager-provider-prop \
     --provider-name JKS \
     --set enabled:true \
     --set include-jvm-default-issuers:true \
     --set trust-store-file:config/truststore
     --set trust-store-pin-file:config/truststore.pin
```

ⓘ **Note:**

If all clients and servers use certificates that are signed by issuers and are included in the JVM's default trust store, you can use the `JVM-Default` trust manager provider to accomplish this task.

Configuring connection handlers
After you configure the key and trust manager providers, update the connection handlers to use the key and trust manager providers.

Steps

- For the LDAP connection handler, use the following command to enable StartTLS with a configuration change. By default, the LDAP connection handler accepts non-secure connections.

  ```
  dsconfig set-connection-handler-prop \
    --handler-name "LDAP Connection Handler" \
    --set allow-start-tls:true \
    --set key-manager-provider:JKS \
    --set trust-manager-provider:JKS \
    --set ssl-cert-nickname:server-cert \
    --set ssl-client-auth-policy:optional
  ```
- If you did not configure secure communication during setup, the LDAPS connection handler is disabled. To configure LDAPS support in this scenario, enable the connection handler and configure most of the same settings. You must set allow-start-tls to `false` and use-ssl to `true`. See the following code for an example configuration.

  ```
  dsconfig set-connection-handler-prop \
    --handler-name "LDAPS Connection Handler" \
    --set enabled:true \
  ```

```
  --set key-manager-provider:JKS \
  --set trust-manager-provider:JKS \
  --set ssl-cert-nickname:server-cert \
  --set ssl-client-auth-policy:optional
```

The following example uses a similar configuration change to enable the HTTPS connection handler.

```
dsconfig set-connection-handler-prop \
  --handler-name "HTTPS Connection Handler" \
  --set enabled:true \
  --set listen-port:443 \
  --set key-manager-provider:JKS \
  --set trust-manager-provider:JKS \
  --set ssl-cert-nickname:server-cert
```

Updating the topology registry

After the server connection handlers are updated to enable TLS, update the topology registry to provide information about the new configuration.

The topology registry holds information about server instances that are part of the environment, and it helps to facilitate inter-server communication, such as replication, mirroring portions of the configuration, and PingDirectory Server's automatic backend server-discovery functionality.

The following table details the two types of entries that require updating.

Configuration types and their update descriptions

Configuration Type	Update description

Server instance listener configuration	▪ Provides information that is needed to trust the TLS certificates that instances in the topology present.
	▪ The server instance listener configuration must include the server certificate, which is defined as the certificate at the head of the chain. This version must be the multi-line, PEM-formatted representation of the certificate. You can use **`dsconfig`** to import the certificate from a file, as shown in the following example.
	`bin/dsconfig set-server-instance-listener-prop \`
	`  --instance-name ds1 \`
	`  --listener-name ldap-listener-mirrored-config \`
	`  --set server-ldap-port:636 \`
	`  --set connection-security:ssl \`
	`  --set 'listener-certificate>/ca/ds1-cert.pem'`
	Note:
	The less-than operator > in the final line indicates that the value is read from a file rather than provided directly. In addition, you might not need to enclose the property name and path within single straight quotes to prevent the shell from interpreting the less-than symbol as an attempt to redirect input.

Server instance configuration	▪ Provides information about options for communicating with those instances.
	▪ Update the server instance configuration object to reflect the new methods that are available for communication with the instance. For example, the `preferred-security` property identifies the mechanism by which other instances in the topology attempt to communicate with the instance.
	The following example code sets the LDAPS and HTTPS ports, indicates that StartTLS support is enabled, and instructs other instances to use SSL (LDAPS) when communicating with the instance.
	`dsconfig set-server-instance-prop \`
	`  --instance-name ds1 \`
	`  --set ldaps-port:636 \`
	`  --set https-port:443 \`
	`  --set preferred-security:ssl \`
	`  --set start-tls-enabled:true`

Troubleshooting TLS-related issues

Use this section for troubleshooting problems that might arise during TLS configuration, including communication and security issues that affect clients as well as PingDirectory Server.

- *Log messages*
- *manage-certificates check-certificate-usability*
- *ldapsearch*
- *Using low-level TLS debugging*

Log messages

The following describes how to use the server's log messages to troubleshoot TLS-related issues.

To troubleshoot TLS-related issues, start by checking the server's access log. If the client can establish a TCP connection to the server, which must occur before TLS negotiation can start, the access log shows a `CONNECT` message with the following information:

- Source and destination address and port for the connection
- Protocol
- Selected client connection policy

The `CONNECT` message does not appear

If the `CONNECT` does not appear, the client might be unable to communicate with the server. The culprit can be a network problem, a firewall that is blocking attempts to communicate, or the client is trying to use an incorrect address or port.

The `CONNECT` message does appear

If the `CONNECT` message appears in the access log, it typically includes a `conn` element that specifies the connection ID. To view additional log messages for the client connection, use the **`search-logs`** tool. For example, if the connection ID is `12345`, the following command displays the complete set of associated log messages.

```
$ bin/search-logs --logFile logs/access conn=12345
```

If you are using LDAPS

If you are attempting to use LDAPS, one of the following log messages appears next:

- `SECURITY-NEGOTIATION` message – Indicates that the client and server successfully completed the negotiation process and that the issue likely occurred after the TLS session was established. This message also includes details about the negotiation, including the TLS protocol and the selected cipher suite.
- `DISCONNECT` message – The issue might involve a failure in the TLS-negotiation process. In such scenarios, the message usually includes a `reason` element that provides additional information about the reason for the disconnect.

  If the failure occurred during TLS negotiation, the usefulness of the `DISCONNECT` message depends in part on whether the failure occurred on the client or the server. For example, if the server decided to abort the negotiation, the message ideally contains the specific reason. If the problem occurred on the client, the log message likely contains only the general category for the failure.

> ⓘ **Note:**
>
> The TLS protocol does not provide a mechanism for conveying detailed error messages. Instead, it offers only a basic alert mechanism with a fixed set of alert types. For example, if a client does not trust the certificate chain that the server presents to it, the server might receive a generic alert like `certificate_unknown`, even if the client knows the precise reason for rejecting the chain. In such

instances, you might need to determine whether the client can provide additional details about the issue.

If the access log does not provide useful information

If the access log does not provide useful information, check the server error log. Although the error log does not normally include information about issues that relate to client communication, it provides helpful information in certain circumstances, like when an internal error within the server interferes with communication attempts.

manage-certificates check-certificate-usability
The **`manage-certificates`** tool offers a **`check-certificate-usability`** subcommand to examine a specified entry in a key store and to identify potential issues that might interfere with secure communication.

The **`check-certificate-usability`** tool completes the following tasks:

- Ensures that a specified entry in the key store includes a private key and a complete certificate chain
- Checks whether the certificate at the root of the chain is found in the Java virtual machine's (JVM's) default set of trusted certificates
- Ensures that the current time lies is within the validity window for all certificates in the chain
- Validates the signatures for all certificates in the chain
- Warns if the end-entity certificate is self-signed
- Warns if the end-entity certificate does not contain an extended key usage extension with the `serverAuth` usage
- Warns if the issuer certificates do not have a key usage extension with the `keyCertSign` usage
- Warns if the issuer certificates do not have a basic constraints extension indicating that it can operate as a certification authority

  If the chain violates a path length constraint, the **`check-certificate-usability`** tool reports an error.
- Ensures that the signature algorithm uses a strong message digest algorithm, like SHA-256

  The **`check-certificate-usability`** tool reports an error for weak digest algorithms like MD5 or SHA-1, and reports a warning for unrecognized digest algorithms.
- Ensures that none of the certificates that use an RSA key pair have a key size less than 2048 bits

The following example demonstrates the usage for the **`manage-certificates check-certificate-usability`** command and its output when no problems are identified.

```
$ bin/manage-certificates check-certificate-usability \
     --keystore config/keystore \
     --keystore-password-file config/keystore.pin \
     --alias server-cert

Successfully retrieved the certificate chain for alias 'server-cert':

Subject DN:  CN=ds1.example.com,O=Example Corp,C=US
Issuer DN:  CN=Example Intermediate CA,O=Example Corp,C=US
Validity Start Time: Tuesday, November 12, 2019 at 03:52:44 PM CST
                     (5 minutes, 45 seconds ago)
Validity End Time: Wednesday, November 11, 2020 at 03:52:44 PM CST
                   (364 days, 23 hours, 54 minutes, 14 seconds from now)
Validity State:  The certificate is currently within the validity window.
Signature Algorithm:  SHA-256 with RSA
Public Key Algorithm:  RSA (2048-bit)
SHA-1 Fingerprint:
 84:e4:00:b9:f0:6b:58:bb:ac:67:79:28:2f:43:9f:e3:ac:24:ee:98
SHA-256 Fingerprint:
 63:85:4d:2c:50:ea:a8:84:54:e0:73:9a:e7:5b:e7:1b:06:85:0e:
```

```
                    28:2b:76:a9:8b:57:fc:27:f7:60:81:48:41

Subject DN:  CN=Example Intermediate CA,O=Example Corp,C=US
Issuer DN:  CN=Example Root CA,O=Example Corp,C=US
Validity Start Time: Tuesday, November 12, 2019 at 03:52:42 PM CST
                     (5 minutes, 47 seconds ago)
Validity End Time: Monday, November 7, 2039 at 03:52:42 PM CST
                   (7299 days, 23 hours, 54 minutes, 12 seconds from now)
Validity State:  The certificate is currently within the validity window.
Signature Algorithm:  SHA-256 with RSA
Public Key Algorithm:  RSA (4096-bit)
SHA-1 Fingerprint:
 de:da:3d:fc:d4:1f:67:79:0a:a1:5a:cd:ca:4a:7e:a5:d3:46:88:27
SHA-256 Fingerprint:

 02:3c:af:ad:b7:07:81:89:45:48:d0:09:31:a8:90:c4:17:11:1c:00:11:fd:49:b2:2c:
   ba:ac:dd:c4:9f:03:36

Subject DN:  CN=Example Root CA,O=Example Corp,C=US
Issuer DN:  CN=Example Root CA,O=Example Corp,C=US
Validity Start Time: Tuesday, November 12, 2019 at 03:52:38 PM CST
                     (5 minutes, 51 seconds ago)
Validity End Time: Monday, November 7, 2039 at 03:52:38 PM CST
                   (7299 days, 23 hours, 54 minutes, 8 seconds from now)
Validity State:  The certificate is currently within the validity window.
Signature Algorithm:  SHA-256 with RSA
Public Key Algorithm:  RSA (4096-bit)
SHA-1 Fingerprint:
 8e:03:e4:58:e6:e3:59:9a:55:77:c0:88:3c:fa:d7:29:f4:ff:de:6c
SHA-256 Fingerprint:
 95:54:0d:e2:aa:48:29:c1:25:7c:20:69:c0:27:33:31:81:07:02:
                    2e:00:24:ae:49:5e:98:bd:a3:72:a5:05:26

OK:  The certificate chain is complete.  Each subsequent certificate is
the issuer for the previous certificate in the chain, and the chain ends
with a self-signed certificate.

OK:  Certificate 'CN=ds1.example.com,O=Example Corp,C=US' has a valid
signature.

OK:  Certificate 'CN=Example Intermediate CA,O=Example Corp,C=US' has a
valid signature.

OK:  Certificate 'CN=Example Root CA,O=Example Corp,C=US' has a valid
signature.

OK:  Certificate 'CN=ds1.example.com,O=Example Corp,C=US' will expire at
Wednesday, November 11, 2020 at 03:52:44 PM CST (364 days, 23 hours, 54
minutes, 14 seconds from now), which is not in the near future.

OK:  Issuer certificate 'CN=Example Intermediate CA,O=Example Corp,C=US'
will expire at Monday, November 7, 2039 at 03:52:42 PM CST (7299 days, 23
hours, 54 minutes, 12 seconds from now), which is not in the near future.

OK:  Issuer certificate 'CN=Example Root CA,O=Example Corp,C=US' will
expire at Monday, November 7, 2039 at 03:52:38 PM CST (7299 days, 23
hours, 54 minutes, 8 seconds from now), which is not in the near future.

OK:  Certificate 'CN=ds1.example.com,O=Example Corp,C=US' at the head of
the chain includes an extended key usage extension, and that extension
includes the serverAuth usage.

OK:  Issuer certificate 'CN=Example Intermediate CA,O=Example Corp,C=US'
includes a basic constraints extension, and the certificate chain
```

```
satisfies those constraints.

OK:  Issuer certificate 'CN=Example Intermediate CA,O=Example Corp,C=US'
includes a key usage extension with the keyCertSign usage flag set to
true.

OK:  Issuer certificate 'CN=Example Root CA,O=Example Corp,C=US' includes
a basic constraints extension, and the certificate chain satisfies those
constraints.

OK:  Issuer certificate 'CN=Example Root CA,O=Example Corp,C=US' includes
a key usage extension with the keyCertSign usage flag set to true.

OK:  Certificate 'CN=ds1.example.com,O=Example Corp,C=US' uses a signature
algorithm of 'SHA-256 with RSA', which is is considered strong.

OK:  Certificate 'CN=Example Intermediate CA,O=Example Corp,C=US' uses a
signature algorithm of 'SHA-256 with RSA', which is is considered strong.

OK:  Certificate 'CN=Example Root CA,O=Example Corp,C=US' uses a signature
algorithm of 'SHA-256 with RSA', which is is considered strong.

OK:  Certificate 'CN=ds1.example.com,O=Example Corp,C=US' has a 2048-bit
RSA public key, which is considered strong.

OK:  Certificate 'CN=Example Intermediate CA,O=Example Corp,C=US' has a
4096-bit RSA public key, which is considered strong.

OK:  Certificate 'CN=Example Root CA,O=Example Corp,C=US' has a 4096-bit
RSA public key, which is considered strong.

No usability errors or warnings were identified while validating the
certificate chain.
```

If any usability issues are identified, they might be responsible for communication problems.

ldapsearch

The **`ldapsearch`** command-line utility is a powerful tool for issuing searches against an LDAP directory server. It also provides a convenient method for troubleshooting a variety of issues, including problems that are relevant to TLS communication.

The following table details arguments that are the most useful for TLS-related communication.

TLS-related communication arguments and their descriptions

Argument	Description
--hostname {address}	Address of the server to which the connection is established
--port {port}	TCP port of the server to which the connection is established. The standard port for non-secure LDAP, or LDAP to be secured with StartTLS, is 389, and the standard port for secure LDAPS is 636. Many deployments use alternate ports, especially non-privileged ports above 1024.
--useSSL	The tool establishes an initially insecure LDAP connection, which is secured later with the StartTLS extended operation.

Argument	Description
--trustStorePath {path}	Path to the trust store that is used when determining whether to trust the certificate chain that the server presents during TLS negotiation. If neither this argument nor the --trustAll argument is provided, the tool prompts you interactively whether to trust server certificates that are not signed by an issuer in the Java virtual machine's (JVM's) default trust store.
--trustStoreFormat {format}	Format for the trust store, which is typically JKS or PKCS12.
--trustStorePassword {password}	Password that is required to access the contents of the trust store.
--trustStorePasswordFile {path}	Path to the file that contains the password that is required to access the contents of the trust store.
--trustAll	The tool blindly trusts all TLS certificate chains that are presented to it. Although this argument can prove useful for troubleshooting purposes, it is not recommended for general use.
--keyStorePath {path}	Path to the key store to use if a client certificate chain is presented to the server. ⓘ **Note:** Use this argument only when one of the following conditions is satisfied: ▪ The server is configured to require clients to present a certificate. ▪ You intend to use the certificate to authenticate through SASL EXTERNAL.
--keyStoreFormat {format}	Format for the key store, which is typically JKS or PKCS12.
--keyStorePassword {password}	Password to access the key store.
--keyStorePasswordFile {path}	Path to the file that contains the password necessary to access the key store.
--certNickname {alias}	Alias of the private key entry in the key store. Use when obtaining the certificate chain to present to the server.
--useSASLExternal	The client authenticates with the EXTERNAL SASL mechanism, which typically identifies the client using the certificate chain that is presented during TLS negotiation.

Argument	Description
--enableSSLDebugging	The tool activates the low-level TLS-debugging feature that the JVM provides.

The following command provides an example of the simplest method for testing TLS communication with PingDirectory Server.

```
$ bin/ldapsearch \
     --hostname ds1.example.com \
     --port 636 \
     --useSSL \
     --baseDN "" \
     --scope base \
     "(objectClass=*)"
The server presented the following certificate chain:

     Subject: CN=ds1.example.com,O=Example Corp,C=US
     Valid From: Tuesday, November 12, 2019 at 08:28:08 PM CST
     Valid Until: Wednesday, November 11, 2020 at 08:28:08 PM
 CST
     SHA-1 Fingerprint:

 6a:22:2a:bd:0b:1b:09:35:63:bc:12:3e:2c:9e:e7:70:bc:a4:73:de
     256-bit SHA-2 Fingerprint:

 7a:8c:e4:76:d4:47:15:fd:65:f5:26:0e:d2:55:77:d7:03:7a:e6:79:9f:bc:
        ae:93:2c:76:9c:01:fc:ef:15:38
     -
     Issuer 1 Subject: CN=Example Intermediate CA,O=Example
 Corp,C=US
     Valid From: Tuesday, November 12, 2019 at 08:28:06 PM CST
     Valid Until: Monday, November 7, 2039 at 08:28:06 PM CST
     SHA-1 Fingerprint:
 01:b3:70:8b:6c:11:43:87:3b:e9:bb:73:27:99:ea:fd:08:c4:db:ec
     256-bit SHA-2 Fingerprint:
 49:60:69:df:33:9d:26:d0:66:c9:6d:7b:0b:cb:3b:96:

 40:22:dc:6d:11:32:b7:c0:30:47:d6:7c:6a:19:cd:60
     -
     Issuer 2 Subject: CN=Example Root CA,O=Example Corp,C=US
     Valid From: Tuesday, November 12, 2019 at 08:28:03 PM CST
     Valid Until: Monday, November 7, 2039 at 08:28:03 PM CST
     SHA-1 Fingerprint:
 b4:83:55:db:82:e4:63:5c:3a:44:13:8f:88:44:e3:60:f2:53:80:48
     256-bit SHA-2 Fingerprint:

 e8:af:6f:ed:b9:0e:df:94:9c:20:29:53:a9:74:44:a9:17:b4:08:65:c8:19:c1:fb:
        34:34:a1:90:83:8a:d5:12

Do you wish to trust this certificate?  Enter 'y' or 'n': y
dn:
objectClass: top
objectClass: ds-root-dse
startupUUID: 8d574122-4584-4522-96d9-0cdcb9d2e339
startTime: 201911130611492

# Result Code:  0 (success)
# Number of Entries Returned:  1
```

Trust stores and trust-related arguments

If no trust-related arguments are provided, the tool uses the JVM's default trust store to verify whether to trust the certificate chain, based on the information that it contains. If a trusted authority has signed the server certificate, the negotiation process continues without further interaction.

If the chain cannot be trusted, based on the information in the JVM-default trust store, **`ldapsearch`** prompts you interactively about whether to trust the certificate. If you accept the chain, the client and server complete the negotiation process, and the client sends the search request to the server. If the search succeeds, the server can communicate over TLS.

To test with a trust store instead of being prompted interactively, use the --trustStorePath argument that points to the appropriate trust store. If you are using a Java Keystore (JKS) trust store, you might not need to provide the trust store password. If you are using a PKCS #12 trust store, you need to provide the trust store password. The following code provides an example.

```
$ bin/ldapsearch \
     --hostname ds1.example.com \
     --port 636 \
     --useSSL \
     --trustStorePath config/truststore.p12 \
     --trustStorePasswordFile config/truststore.pin \
     --trustStoreFormat PKCS12 \
     --baseDN "" \
     --scope base \
     "(objectClass=*)"
dn:
objectClass: top
objectClass: ds-root-dse
startupUUID: c8724159-8c37-45eb-b210-879bfcf74ad6
startTime: 20191113154023Z

# Result Code:  0 (success)
# Number of Entries Returned:  1
```

Client certificate chains and key stores

To present a client certificate chain to the server, either because the server's connection handler is configured with an ssl-client-auth-policy value of required or because you plan to use the certificate to authenticate by way of the SASL EXTERNAL mechanism, provide at least the key store and its corresponding password. You can also specify the alias of the certificate chain to present, which is recommended if your client key store contains multiple certificates. The following code provides an example.

```
$ bin/ldapsearch \
     --hostname ds1.example.com \
     --port 636 \
     --useSSL \
     --trustStorePath config/truststore.p12 \
     --trustStorePasswordFile config/truststore.pin \
     --trustStoreFormat PKCS12 \
     --keyStorePath client-keystore \
     --keyStorePasswordFile client-keystore.pin \
     --certNickname client-cert \
     --useSASLExternal \
     --baseDN "" \
     --scope base \
     "(objectClass=*)"
```

```
dn:
objectClass: top
objectClass: ds-root-dse
startupUUID: c8724159-8c37-45eb-b210-879bfcf74ad6
startTime: 20191113154023Z

# Result Code:  0 (success)
# Number of Entries Returned:  1
```

If you need to further troubleshoot a TLS-related issue

If you encounter a TLS-related issue that you cannot resolve by examining the **`ldapsearch`** output or the server logs, use the --enableSSLDebugging option to enable the JVM's support for low-level debugging of TLS processing. For more information, see *Using low-level TLS debugging* on page 360.

Using low-level TLS debugging

Use tools other than the command-line tools that are provided with PingDirectory Server for performing low-level TLS debugging.

Before you begin

> ⓘ **Note:**
>
> If you need to use low-level debugging options, enable the Java Virtual Machine (JVM)'s support for TLS debugging. Many of the command-line tools that are provided with PingDirectory Server, such as **`ldapsearch`**, offer an --enableSSLDebugging argument that simplifies this process.

Steps

1. In the `config/java.properties` file, add the following line to the set of properties for the appropriate tool.

   ```
   -Djavax.net.debug=all
   ```

2. For the changes to take effect, run the **`bin/dsjavaproperties`** command.

Next steps

The next time the tool is run, an output is generated detailing the TLS-related processing that the JVM is performing. You and the support team can use the output to identify the issue.

Enabling low-level debugging

Before you begin

> ⓘ **Note:**
>
> Because this approach requires multiple server restarts, it is not a popular option. However, you might be able to obtain more information without a restart by using the debugging support that is built into the server. For more information and to enable this level of support, see *Using the debug log publisher*.

Steps

1. In the `config/java.properties` file, add the following line to the `start-server.java-args` property.

```
-Djavax.net.debug=all
```

2. Run **`bin/dsjavaproperties`**.
3. Restart the server.

Using the debug log publisher

About this task
Use the debugging support that is built into the server to obtain more information without a restart.

Steps

1. To enable the debug log publisher, run the following configuration changes.

```
dsconfig create-debug-target \
     --publisher-name "File-Based Debug Logger" \
     --target-name
 com.unboundid.directory.server.extensions.TLSConnectionSecurityProvider \
     --set debug-level:verbose

dsconfig set-log-publisher-prop \
     --publisher-name "File-Based Debug Logger" \
     --set enabled:true
```

   After you make these changes, the `logs/debug` file captures a substantial amount of information about the TLS-related processing that the server is performing. Although this file does not provide as much detail as the JVM's built-in debugging information, it might help to pinpoint the cause of the problem and to identify potential solutions.
2. When you no longer require this level of debugging, disable the debug log publisher and remove the debug target.

```
dsconfig set-log-publisher-prop \
     --publisher-name "File-Based Debug Logger" \
     --set enabled:false

dsconfig delete-debug-target \
     --publisher-name "File-Based Debug Logger" \
     --target-name
 com.unboundid.directory.server.extensions.TLSConnectionSecurityProvider
```

> ⓘ **Tip:**
>
> To troubleshoot TLS communication with a non-Java client that does not offer its own TLS debugging mechanism, and if the server-side debugging support is insufficient, use a network protocol analyzer to capture the communication between the client and the server, and to examine its content. The free open-source *Wireshark utility* is an excellent graphical tool that runs on a variety of platforms and provides excellent support for understanding TLS communication. Even if you cannot decipher the encrypted content, you can view at least some of the handshake messages. Unfortunately, more of the handshake is encrypted in TLS 1.3 than in earlier versions of the protocol. Although this change improves security and privacy, it might interfere with troubleshooting attempts.

Configuring load balancing

You can distribute the load on your Directory Proxy Server using one of the load-balancing algorithms provided with PingDirectoryProxy Server. By default, the Directory Proxy Server prefers local servers over non-local servers, unless you set the `use-location` property of the load-balancing algorithm to false. Within a given location, the Directory Proxy Server prefers available servers over degraded servers. This means that if at all possible, the Directory Proxy Server sends requests to servers that are local and available before considering selecting any server that is non-local or degraded.

ⓘ **Note:** If the `use-location` property is set to true, then the load is balanced only among available external servers in the same location. If no external servers are available in the same location, the Directory Proxy Server will attempt to use available servers in the first preferred failover location, and so on. The failover based on no external servers with AVAILABLE health state can be customized to allow the Directory Proxy Server to prefer local DEGRADED health servers to servers in a failover location. See the online *PingDirectoryProxy Server Configuration Reference Guide* for more information on the `prefer-degraded-servers-over-failover` property.

The Directory Proxy Server provides the following load-balancing algorithms:

- **Failover load balancing**. This algorithm forwards requests to servers in a given order, optionally taking the location into account. If the preferred server is not available, then it will fail over to the alternate server in a predefined order. This balancing method can be useful if certain operations, such as LDAP writes, need to be forwarded to a primary external server, with secondary external servers defined for failover if necessary.

  This algorithm also offers load spreading to multiple failover servers. If the failover load-balancing algorithm is configured with one or more load-spreading base DNs, then requests that target entries below a load-spreading base DN can be balanced across any of the servers with the same health check state and location. Requests targeting a specific portion of the data will consistently be routed to the same server, but requests targeting a different portion of the data may be sent to a different server.
- **Fewest operations load balancing**. This algorithm forwards requests to the backend server with the fewest operations currently in progress and tends to exhibit the best performance.
- **Health weighted load balancing**. This algorithm assigns weights to servers based on their health scores and, optionally, their locations. For example, servers with a higher health check score will receive a higher proportion of the requests than servers with lower health check scores.
- **Single server load balancing**. This algorithm forwards all operations to a single external server that you specify.
- **Weighted load balancing**. This algorithm uses statically defined weights for sets of servers to divide load among external servers. External servers are grouped into weighted sets, the values of which, when added to all of the weighted sets for the load balancing algorithm, represent a percentage of the load the external servers should receive.
- **Criteria based load balancing**. This algorithm allows you to balance your load across a server topology depending on the types of operations received or the client issuing the request.

  For example, ds1 and ds2 are assigned to a weighted set named Set-80 and assigned the weight 80. The external servers ds3 and ds4 are assigned to the weighted set Set-20 and assigned the weight 20. When both sets, Set-80 and Set-20, are assigned to the load balancing algorithm, 80 percent of the load will be forwarded to ds1 and ds2, while the remaining 20 percent will be forwarded to ds3 and ds4.

Configure failover load-balancing for load spreading

If the failover load-balancing algorithm is configured with one or more load-spreading base DNs, then requests that target entries below a load-spreading base DN may be balanced across any of the servers with the same health check state and location. Requests targeting a specific portion of the data will consistently be routed to the same server, but requests targeting a different portion of the data may be sent to a different server.

Load spreading is useful for deployments in which the DIT contains a large number of branches below a common parent, and in which most operations (including search operations, as indicated by the search base DN) only target entries at least one level below that common parent. For example, this may be useful for a multi-tenant deployment in which all of the entries for a given tenant are within their own branch, and all of the tenant branches reside below a common parent.

Load spreading is configured with the **`load-spreading-base-dn`** property. The value(s) of this property are the base DN(s) below which the tenant entries reside. For example, in a deployment with a DIT like the following, the **`load-spreading-base-dn`** value would be set to **`ou=customers,dc=example,dc=com`**:

- **`dc=example,dc=com`**
  - **`ou=customers,dc=example,dc=com`**
    - **`ou=Customer 1,ou=customers,dc=example,dc=com`**
    - **`ou=Customer 2,ou=customers,dc=example,dc=com`**
    - **`ou=Customer 3,ou=customers,dc=example,dc=com`**
    - ...

If the **`load-spreading-base-dn`** property is not configured, the failover load-balancing algorithm will uses the default behavior. If the property is configured with one or more values, but a client requests an operation that targets an entry that is not below any of the configured base DNs, then that operation will be handled using the default behavior. When the **`load-spreading-base-dn`** property is configured with one or more values, the load-balancing algorithm will continue to generate the same list of lists, but the order of the servers within each list will be determined using the following algorithm:

1. If the list is empty or contains only a single item, then leave it unchanged and skip the remaining steps.
2. Identify the RDN component from the target entry DN that is exactly one level below one of the **`load-spreading-base-dn`** values. If the targeted entry is not below any of the configured **`load-spreading-base-dn`** values, then the order of servers in each of those lists will be based only on the order in which they appear in the load-balancing algorithm's **`backend-server`** property. The remaining steps are skipped.
3. Compute a SHA-1 digest from the normalized string representation of the identified RDN component. SHA-1 is notably faster than more secure digest algorithms, and it does a very good job at distributing bits across the entire range of the 160 bits that it generates.
4. Create a non-negative integer from the last 31 bits of the computed SHA-1 digest.
5. Compute a modulus using the integer value as the dividend, and the number of servers in the current list as the divisor. This will yield an integer value that is between 0 and (list.size() - 1), inclusive.
6. If the modulus computed is equal to zero, no further action is necessary. If not, move a number of servers equal to the computed modulus from the beginning of the list to the end of the list. The order of the elements that are moved should be preserved.

For example, consider a **`load-spreading-base-dn`** value of **`"ou=customers,dc=example,dc=com"`**, a list that contains three servers (ds1, ds2, and ds3, in that order), and a modify request that targets the entry with DN **`"uid=jdoe,ou=People,ou=Acme,ou=customers,dc=example,dc=com"`**. The RDN component immediately below the **`load-spreading-base-dn`** is **`"ou=Acme"`**. The normalized string representation of that RDN component is **`"ou=acme"`**, and the hexadecimal representation of the SHA-1 digest of that is **`"f0c69713535daf8816038f1bceab70380c92b83e"`**. The last 31 bits of that SHA-1 digest are 0c92b83e hex, which is 210942014. With 210942014 modulo 3 is 2, which means that the first two servers are moved from the beginning of the list to the end of the list, resulting in and order of ds3, ds1, ds2.

While this algorithm will spread the load across multiple backend servers, it does not mean that there will be an even distribution of the load across all of those servers. The load-balancing algorithm will still prioritize based on location and health check state, so the load will generally be spread only across the available servers in the same location as the Directory Proxy Server. Second, assuming that the entries that are immediate children of a **`load-spreading-base-dn`** are the tops of the branches that define tenants, some tenants will still be targeted more heavily than others (because they have more entries, or

because their entries are accessed more frequently). The modulo operation may therefore not result in an even distribution across those servers.

Configuring load balancing using dsconfig

Steps

1. Use the **`dsconfig`** tool to create and configure a load-balancing algorithm.

   ```
   $ bin/dsconfig
   ```

   Specify the host name, connection method, port number, and bind DN as described in previous procedures.
2. In the Directory Proxy Server main menu, enter the number associated with load-balancing algorithms.
3. Select an existing load-balancing algorithm to use as a template or select `n` to create a new load-balancing algorithm from scratch.

   ```
   >>>>Choose how to create the new Load Balancing Algorithm:

   n) new Load Balancing Algorithm created from scratch
   t) use an existing Load Balancing Algorithm as a template
   b) back
   q) quit

   Enter a choice [n]: n
   ```
4. Select the type of load-balancing algorithm that you want to create. Depending on type of algorithm you select, you will be guided through a series of configuration properties, such as providing a name and selecting an LDAP external server.

   ```
   >>>> Select the type of Load Balancing Algorithm that you want to
   create:

       1)  Failover Load Balancing Algorithm
       2)  Fewest Operations Load Balancing Algorithm
       3)  Health Weighted Load Balancing Algorithm
       4)  Single Server Load Balancing Algorithm
       5)  Weighted Load Balancing Algorithm

       ?)  help
       c)  cancel
       q)  quit

   Enter choice [c]: 3
   ```
5. Review the configuration properties for your new load-balancing algorithm. If you are satisfied, enter `f` to finish.

Configuring criteria-based load-balancing algorithms

You can configure alternate load-balancing algorithms that determine how they function according to request or connection criteria. These algorithms allow you to balance your load across a server topology depending on the types of operations received or the client issuing the request. They are called criteria-based load-balancing algorithms and are configured using at least one connection criteria or request criteria. For example, you can configure criteria-based load-balancing algorithms to accomplish the following:

- Route write operations to a single server from a set of replicated servers, to prevent replication conflicts, while load balancing all other operations across the full set of servers.
- Route all operations from a specific client to a single server in a set of replicated servers, eliminating errors that arise from replication latency, while load balancing operations from other clients across the

full set of servers. This configuration is useful for certain provisioning applications that need to write and then immediately read the same data.

When a request is received, the proxying request processor first iterates through all of the criteria-based load-balancing algorithms in the order in which they are listed, to determine whether the request matches the associated criteria. If there is a match, then the criteria-based load-balancing algorithm is selected. If there is not a match, then the default load-balancing algorithm is used.

Preferring failover LBA for write operations

An administrator can configure the Directory Proxy Server to use Criteria-Based Load-Balancing Algorithms to strike a balance between providing a consistent view of directory server data for applications that require it and taking advantage of all servers in a topology for handling read-only operations, such as search and bind. The flexible configuration model supports a wide range of criteria for choosing which Load-Balancing Algorithm to use for each operation. In most Directory Proxy Server deployments, using a Failover Load-Balancing Algorithm for at least ADD, DELETE, and MODIFY-DN operations if not for all types of write operations is recommended.

Each Proxying Request Processor configured in the Directory Proxy Server uses a Load-Balancing Algorithm to choose which Directory Server to use for a particular operation. The Load-Balancing Algorithm takes several factors into account when choosing a server:

- The availability of the directory servers.
- The location of the directory servers. By default Load-Balancing Algorithms prefer directory servers in the same location as the Directory Proxy Server.
- Whether the Directory Server is degraded for any reason, such as having a Local DB Index being rebuilt.
- The result of configured Health Checks. For instance, a server with a small replication backlog can be preferred over one with a larger backlog.
- Recent operation routing history.

How these factors are used depends on the specific Load-Balancing Algorithm. The two most commonly used Load-Balancing Algorithms are the Failover Load-Balancing Algorithm and the Fewest Operations Load-Balancing Algorithm. These two algorithms are similar when determining which Directory Servers are the possible candidates for a specific operation. The algorithms use the same criteria to determine server availability and health, and by default they will prefer Directory Servers in the same location as the Directory Proxy Server. However, they differ in the criteria they use to choose between available servers.

The Failover Load-Balancing Algorithm will send all operations to a single server until it is unavailable, and then it will send all operations to the next preferred server, and so on. This algorithm provides the most consistent view of the topology to clients because all clients (at least those in the same location as the Directory Proxy Server) will see the same, up-to-date view of the data, but it leaves unused capacity in the failover instances since most topologies include multiple Directory Server replicas within each data center.

On the other hand, the Fewest Operations Load-Balancing Algorithm does the best job of efficiently distributing traffic among multiple servers since it chooses to send each operation to the server that has the fewest number of outstanding operations--that is, the server from the Directory Proxy Server's point of view that is the least busy. (Note: the Fewest Operations Load-Balancing Algorithm routes traffic to the least loaded server, which in a lightly-loaded environment can result in an imbalance since the first server in the list of configured servers is more likely to receive a request.) This algorithm naturally routes to servers that are more responsive as well as limiting the impact of servers that have become unreachable. However, this implies that consecutive operations that depend on each other can be routed to different Directory Servers, which can cause issues for some types of clients:

- If two entries are added in quick succession where the first entry is the parent of the second in the LDAP hierarchy, then the addition of the child entry could fail if that operation is routed to a different Directory Server instance than the first ADD operation, and this happens within the replication latency.
- Some clients add or modify an entry and then immediately read the entry back from the server, expecting to see the updates reflected in the entry.

In these situations, it is desirable to configure the <keyword keyref="PROXY_SERVER_BASE_NAME"/> to route dependent requests to the same server.

The server affinity feature (see *Configuring Server Affinity*) achieves this in some environments but not in all because the affinity is tracked independently by each Directory Proxy Server instance, and some clients send requests to multiple proxies. It is common for a client to not connect to the Directory Proxy Servers directly but instead to connect through a network load balancer, which in turn opens connections to the Directory Proxy Servers. Each individual client connection will be established to a single Directory Proxy Server so that operations on that connection will be routed to the same Directory Proxy Server, and server affinity configured within the Directory Proxy Server will ensure those operations will be routed to the same Directory Server. However, many clients establish a pool of connections that are reused across operations, and within this pool, connections will be established through the load balancer to different Directory Proxy Servers. Dependent operations sent on different connections could then be routed to different Directory Proxy Servers, and then on to different Directory Servers.

A Failover Load-Balancing Algorithm addresses this issue by routing all requests to a single server, but that leaves unused search capacity on the other instances. A Criteria Based Load-Balancing Algorithm enables the proxy to route certain types of requests (or requests from certain clients) using a different Load-Balancing Algorithm than the default. For instance, all write operations (i.e., ADD, DELETE, MODIFY, and MODIFY-DN) could be routed using a Failover Load-Balancing Algorithm, while all other operations (bind, search, and compare) use a Fewest Operations Load-Balancing Algorithm. And in addition, if there are clients that are particularly sensitive to reading entries immediately after modifying them, additional Connection Criteria can be specified to all operations from those clients using the Failover Load-Balancing Algorithm. Note that, routing all write requests to a single server in a location instead of evenly across servers does not limit the overall throughput of the system since all servers ultimately have to process all write operations either from the client directly or via replication.

Another benefit of using the Failover Load-Balancing Algorithm for write operations is reducing replication conflicts. The Ping Identity Directory Server follows the traditional LDAP replication model of eventual consistency. This provides very high availability for handling write traffic even in the presence of network partitions, but it can lead to replication conflicts. Replication conflicts involving modify operations can be automatically resolved, leaving the servers in a consistent state where each attribute on each entry reflects the most recent update to that attribute. However, conflicts involving ADD, DELETE, and MODIFY-DN operations cannot always be resolved automatically and can require manual involvement from an administrator. By routing all write operations (or at least ADD, DELETE, and MODIFY-DN operations) to a single server, replication conflicts can be avoided.

There are a few points to consider when using a Failover Load-Balancing Algorithm:

- When using the Failover Load-Balancing Algorithm in a configuration with multiple locations, the Load-Balancing Algorithm will fail over between local instances before failing over to servers in a remote location. The list of servers in the `backend-server` configuration property of the Load-Balancing Algorithm should be ordered such that preferred local servers should appear before failover local servers, but the relative order of servers in different locations is unimportant as the `preferred-failover-location` of the Directory Proxy Server's configuration is used to decide which remote location to fail over to. It is also advisable that the order of local servers match the `gateway-priority` configuration settings of the "Replication Server" configuration object on the Directory Server instances. This can reduce the WAN replication delay because the Directory Proxy Server will then prefer to send writes to the Directory Server with the WAN Gateway role, avoiding an extra hop to the remote locations.
- For Directory Proxy Server configurations that include multiple Proxying Request Processors, including Entry-Balancing environments, each Proxying Request Processor should be updated to include its own Criteria-Based Load-Balancing Algorithm.

Routing operations to a single server

About this task

The following example shows how to extend a Directory Proxy Server's configuration to use a Criteria Based Load Balancing Algorithm to route all write requests to a single server using a Failover Load Balancing Algorithm. The approach outlined here can easily be extended to support alternate criteria as well as more complex topologies using multiple locations or Entry Balancing.

This example uses a simple deployment of a Directory Proxy Server fronting three Directory Servers: ds1.example.com, ds2.example.com, and ds3.example.com.

Once these configurations changes are applied, the Directory Proxy Server will route all write operations to ds1.example.com as long as it is available and then to ds2.example.com if it is not, while routing other types of operations, such as searches and binds, to all three servers using the Fewest Operations Load Balancing Algorithm.

Steps

1. First, create a location.

```
dsconfig create-location --location-name Austin
```

2. Update the failover location for your server.

```
dsconfig set-location-prop --location-name Austin
```

3. Set the location as a global configuration property.

```
dsconfig set-global-configuration-prop --set location:Austin
```

4. Set up the health checks for each external server.

```
dsconfig create-ldap-health-check \
--check-name ds1.example.com:389_dc_example_dc_com-search-health-check \
--type search --set enabled:true --set base-dn:dc=example,dc=com

dsconfig create-ldap-health-check \
--check-name ds2.example.com:389_dc_example_dc_com-search-health-check \
--type search --set enabled:true --set base-dn:dc=example,dc=com

dsconfig create-ldap-health-check \
--check-name ds3.example.com:389_dc_example_dc_com-search-health-check \
--type search --set enabled:true --set base-dn:dc=example,dc=com
```

5. Create the external servers.

```
dsconfig create-external-server --server-name ds1.example.com:389 \
--type Ping Identity-ds \
--set server-host-name:ds1.example.com --set server-port:389 \
--set location:Austin --set "bind-dn:cn=Proxy User,cn=Root DNs,cn=config" \
--set password:AADoPkhx22qpiBQJ7T0X4wH7 \
--set health-check:ds1.example.com:389_dc_example_dc_com-search-health-check


dsconfig create-external-server --server-name ds2.example.com:389 \
--type Ping Identity-ds \
--set server-host-name:ds2.example.com --set server-port:389 \
--set location:Austin --set "bind-dn:cn=Proxy User,cn=Root DNs,cn=config" \
--set password:AAAoVqVYsEavey80T0QfR60I \
--set health-check:ds2.example.com:389_dc_example_dc_com-search-health-check


dsconfig create-external-server --server-name ds3.example.com:389 \
```

```
--type Ping Identity-ds \
--set server-host-name:ds3.example.com --set server-port:389 \
--set location:Austin --set "bind-dn:cn=Proxy User,cn=Root DNs,cn=config" \
--set password:AADOkveb0TtYR9xpkVrNgMtF \
--set health-check:ds3.example.com:389_dc_example_dc_com-search-health-check
```

6. Create a Load Balancing Algorithm for `dc=example,dc=com`.

```
dsconfig create-load-balancing-algorithm \
--algorithm-name dc_example_dc_com-fewest-operations \
--type fewest-operations --set  enabled:true \
--set backend-server:ds1.example.com:389 \
--set backend-server:ds2.example.com:389 \
--set backend-server:ds3.example.com:389
```

7. Create a Request Processor for `dc=example,dc=com`.

```
dsconfig create-request-processor \
--processor-name dc_example_dc_com-req-processor \
--type proxying \
--set load-balancing-algorithm:dc_example_dc_com-fewest-operations
```

8. Create a Subtree View for `dc=example,dc=com`.

```
dsconfig create-subtree-view \
--view-name dc_example_dc_com-view \
--set base-dn:dc=example,dc=com \
--set request-processor:dc_example_dc_com-req-processor
```

9. Update the client connection policy for dc=example,dc=com.

```
dsconfig set-client-connection-policy-prop \
--policy-name default \
--add subtree-view:dc_example_dc_com-view
```

10. Create a new Request Criteria object to match all write operations.

```
dsconfig create-request-criteria \
--criteria-name any-write \
--type simple --set "description:All Write Operations" \
--set operation-type:add --set operation-type:delete \
--set operation-type:modify --set operation-type:modify-dn
```

11. Create a new Failover Load Balancing Algorithm listing the servers that should be included. Note the order that the servers are listed here is the failover order between servers.

```
dsconfig create-load-balancing-algorithm \
--algorithm-name dc_example_dc_com-failover \
--type failover --set enabled:true \
--set backend-server:ds1.example.com:389 \
--set backend-server:ds2.example.com:389 \
--set backend-server:ds3.example.com:389
```

12. Tie the Request Criteria and the Failover Load Balancing Algorithm together into a Criteria Based Load Balancing Algorithm.

```
dsconfig create-criteria-based-load-balancing-algorithm \
--algorithm-name dc_example_dc_com-write-traffic-lba \
--set "description:Failover LBA For All Write Traffic" \
--set request-criteria:any-write \
--set load-balancing-algorithm:dc_example_dc_com-failover
```

13. Update the Proxying Request Processor to use the Criteria Based Load Balancing Algorithm.

```
dsconfig set-request-processor-prop \
--processor-name dc_example_dc_com-req-processor \
```

```
--set criteria-based-load-balancing-algorithm:dc_example_dc_com-write-
traffic-lba
```

Routing operations from a single client to a specific set of servers

About this task

To create a type of server affinity, where all operations from a single client are routed to a specific set of servers, follow a similar process as in the previous use case. Instead of request criteria, configure connection criteria. These connection criteria identify clients that could be adversely affected by replication latency. These clients will use the Failover Load Balancing Algorithm rather than the default Fewest Operations Load Balancing Algorithm.

For example, an administrative tool includes a "delete user" function. If the application immediately re-queries the directory for an updated list of users, the just-deleted entry must not be included. To configure a criteria-based load balancing algorithm to support this use case, perform the following:

- Create a failover load balancing algorithm that lists the same set of servers as the existing fewest operation load balancing algorithm.
- Create connection criteria that match the clients for which failover load balancing should be applied, rather than fewest operations load balancing.
- Create a criteria-based load balancing algorithm that references the two configuration objects created in the previous steps.
- Assign the new load balancing algorithm to the proxying request processor.

The following procedure provides examples of each of these steps.

Steps

1. Create the new failover load balancing algorithm using **`dsconfig`** as follows:

```
dsconfig create-load-balancing-algorithm \
  --algorithm-name client_one_routing_algorithm \
  --type failover --set enabled:true \
  --set backend-server:east1.example.com:389 \
  --set backend-server:east2.example.com:389
```

2. To route operations from a single client to a single server in a set of replicated servers, create connection criteria using **`dsconfig`** as follows:

```
dsconfig create-connection-criteria \
  --criteria-name "Client One"  --type simple \
  --set included-user-base-dn:cn=Client One,ou=Apps,dc=example,dc=com
```

3. Configure a criteria-based load balancing algorithm and assign it to the proxying request processor. Use the load balancing algorithm and connection criteria created in the previous steps:

```
dsconfig create-criteria-based-load-balancing-algorithm \
  --algorithm-name dc_example_dc_com-client-operations \
  --set load-balancing-algorithm:dc_example_dc_com-failover \
  --set "request-criteria:Client One Requests"
```

4. Assign the new criteria-based load balancing algorithm to the proxying request processor using **`dsconfig`** as follows:

```
dsconfig set-request-processor-prop \
  --processor-name dc_example_dc_com-req-processor \
  --add criteria-based-load-balancing-agorithm:dc_example_dc_com-client-
operations
```

Understanding failover and recovery

Once a previously degraded or unavailable server has recovered, it should be eligible to start receiving traffic within the time configured for the health-check-frequency property, 30 seconds by default. However, failover and recovery also depend on the load-balancing algorithm in use.

The load-balancing algorithm provides an ordered list of servers to check, with the number of servers in the list based on the maximum number of retry attempts. The server checks to see if affinity should be used and, if so, whether an affinity is set for that load-balancing algorithm. If there is an affinity to a particular server and that server is classified as available, then that server will always be the first in the list.

Next, the Directory Proxy Server creates a two-dimensional matrix of servers based on the health check state (with available preferred over degraded and unavailable not considered at all) and location (with backend servers in the same location as the Directory Proxy Server most preferred, then servers in the first failover location, then the second, and so on). Within each of these sets, and ideally at least one server in the local data center is classified as available, the load-balancing algorithm selects the servers in the order of most preferred to least preferred based on whatever logic the load-balancing algorithm uses. The load-balancing algorithm keeps selecting servers until enough of them have been selected to satisfy the maximum number of possible retries.

The load-balancing algorithm includes a configuration option that allows you to decide whether to prefer location over availability and vice-versa. For example, is a local degraded server more or less preferred than a remote available server? By default, the algorithm will prefer available servers over degraded ones, even if it has to go to another data center to access them. You can change the load-balancing algorithm to try to stay in the same data center if at least one server is not unavailable.

The Directory Proxy Server does both proactive and reactive health checking. With proactive health checking, the Directory Proxy Server will periodically (by default, every 30 seconds), run a full set of tests against each backend server. The result of these tests will be usd to determine the overall health check state (available, degraded, or unavailable) and score (and integer value from 10 to 0). With reactive health checking, the Directory Proxy Server may kick off a lesser set of health checks against a server if an operation forwarded to that server did not complete successfully.

Proactive health checking can be used to promote and demote the health of a server, but reactive health checking can only be used to demote the health of a server. As a result, if a server is determined to be unavailable, then it will remain that way until a subsequence proactive health check determines that it has recovered. If a server is determined to be degraded, it may not become available until the next proactive health check, but it could be downgraded to unavailable by a reactive check if other failures are encountered against that server.

Both proactive and reactive health check assignments take effect immediately and will be considered for all subsequent requests routed to the load-balancing algorithm. If a a server is considered degraded, then it will immediately be considered less desirable than available servers in the same data center, and possibly less desirable than available servers in more remote data centers. If a server is considered unavailable, then it will not be eligible to be selected until it is reclassified as available or degraded.

Configuring HTTP connection handlers

HTTP connection handlers are responsible for managing the communication with HTTP clients and invoking servlets to process requests from those clients. They can also be used to host web applications on the server. Each HTTP connection handler must be configured with one or more HTTP servlet extensions and zero or more HTTP operation log publishers.

If the HTTP Connection Handler cannot be started (for example, if its associated HTTP Servlet Extension fails to initialize), then this will not prevent the entire Directory Proxy Server from starting. The Directory Proxy Server's **`start-server`** tool will output any errors to the error log. This allows the Directory Proxy Server to continue serving LDAP requests even with a bad servlet extension.

The configuration properties available for use with an HTTP connection handler include:

- **listen-address**. Specifies the address on which the connection handler will listen for requests from clients. If not specified, then requests will be accepted on all addresses bound to the system.
- **listen-port**. Specifies the port on which the connection handler will listen for requests from clients. Required.
- **use-ssl**. Indicates whether the connection handler will use SSL/TLS to secure communications with clients (whether it uses HTTPS rather than HTTP). If SSL is enabled, then `key-manager-provider` and `trust-manager-provider` values must also be specified.
- **http-servlet-extension**. Specifies the set of servlet extensions that will be enabled for use with the connection handler. You can have multiple HTTP connection handlers (listening on different address/port combinations) with identical or different sets of servlet extensions. At least one servlet extension must be configured.
- **http-operation-log-publisher**. Specifies the set of HTTP operation log publishers that should be used with the connection handler. By default, no HTTP operation log publishers will be used.
- **key-manager-provider**. Specifies the key manager provider that will be used to obtain the certificate presented to clients if SSL is enabled.
- **trust-manager-provider**. Specifies the trust manager provider that will be used to determine whether to accept any client certificates presented to the server.
- **num-request-handlers**. Specifies the number of threads that should be used to process requests from HTTP clients. These threads are separate from the worker threads used to process other kinds of requests. The default value of zero means the number of threads will be automatically selected based on the number of CPUs available to the JVM.
- **web-application-extension**. Specifies the Web applications to be hosted by the server.

Configuring an HTTP connection handler

About this task

An HTTP connection handler has two dependent configuration objects: one or more HTTP servlet extensions and optionally, an HTTP log publisher. The HTTP servlet extension and log publisher must be configured prior to configuring the HTTP connection handler. The log publisher is optional but in most cases, you want to configure one or more logs to troubleshoot any issues with your HTTP connection.

Steps

1. The first step is to configure your HTTP servlet extensions. The following example uses the ExampleHTTPServletExtension in the Server SDK.

```
$ bin/dsconfig create-http-servlet-extension \
  --extension-name "Hello World Servlet" \
  --type third-party \
  --set "extension-
class:com.unboundid.directory.sdk.examples.ExampleHTTPServletExtension" \
  --set "extension-argument:path=/" \
  --set "extension-argument:name=example-servlet"
```

2. Next, configure one or more HTTP log publishers. The following example configures two log publishers: one for common access; the other, detailed access. Both log publishers use the default configuration settings for log rotation and retention.

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

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

3. Configure the HTTP connection handler by specifying the HTTP servlet extension and log publishers. Note that some configuration properties can be later updated on the fly while others, like `listen-port`, require that the HTTP Connection Handler be disabled, then re-enabled for the change to take effect.

```
$ bin/dsconfig create-connection-handler \
  --handler-name "Hello World HTTP Connection Handler" \
  --type http \
  --set enabled:true \
  --set listen-port:8443 \
  --set use-ssl:true \
  --set "http-servlet-extension:Hello World Servlet" \
  --set "http-operation-log-publisher:HTTP Common Access Logger" \
  --set "http-operation-log-publisher:HTTP Detailed Access Logger" \
  --set "key-manager-provider:JKS" \
  --set "trust-manager-provider:JKS"
```

4. By default, the HTTP connection handler has an advanced monitor entry property, `keep-stats`, that is set to TRUE by default. You can monitor the connection handler using the **`ldapsearch`** tool.

```
$ bin/ldapsearch --baseDN "cn=monitor" \
  "(objectClass=ds-http-connection-handler-statistics-monitor-entry)"
```

HTTP correlation IDs

Correlation IDs make it easier to track log messages across a software system request that passes through multiple subsystems.

Scattered and intermingled log messages create challenges for tracing the request flow on distributed systems. A correlation ID can be assigned to a request and added to all associated operations as the request is processed. A correlation ID allows related log messages to be easily located and grouped. The server supports correlation IDs for all HTTP requests received through its HTTP or HTTPS Connection Handler.

When an HTTP request is received, it is automatically assigned a correlation ID. This ID can be used to correlate HTTP responses with messages recorded to the HTTP Detailed Operation log and the trace log. For specific web APIs, the correlation ID might also be passed to the LDAP subsystem.

The correlation ID appears with associated requests in LDAP logs in the **`correlationID`** key for the following REST APIs:

- SCIM 1
- Delegated admin
- Consent
- Directory

The correlation ID is used as the default client request ID value in intermediate client request controls used by the following REST APIs:

- SCIM 2
- Consent
- Directory

Values related to the intermediate client request control appear in the LDAP logs in the **via** key and are forwarded to downstream LDAP servers when received by PingDirectoryProxy Server. The correlation ID header is also added to requests forwarded by the PingDataGovernance gateway.

For Server SDK extensions that have access to the current `HttpServletRequest`, the current correlation ID can be retrieved as a string through the `HttpServletRequest` **com.pingidentity.pingdata.correlation_id** attribute, as shown.

```
(String) request.getAttribute("com.pingidentity.pingdata.correlation_id");
```

Configuring HTTP correlation ID support

About this task

Correlation ID support is enabled by default for each HTTP Connection Handler.

Steps

- To enable or disable correlation ID support for the HTTPS Connection Handler, use the `set-connection-handler-prop` option with **dsconfig**.

  This example shows how to enable correlation ID support.

  ```
  $ dsconfig set-connection-handler-prop \
    --handler-name "HTTPS Connection Handler" \
    --set use-correlation-id-header:true
  ```

  This example shows how to disable correlation ID support.

  ```
  $ dsconfig set-connection-handler-prop \
    --handler-name "HTTPS Connection Handler" \
    --set use-correlation-id-header:false
  ```

- To customize the response header name for the correlation ID, use the `set-connection-handler-prop` option with **dsconfig**.

  > ⓘ **Note:**
  >
  > The server generates a correlation ID for every HTTP request and sends it in the response through the **Correlation-Id** response header.

  This example changes the **correlation-id-response-header** property value to `X-Request-Id`.

  ```
  $ dsconfig set-connection-handler-prop \
     --handler-name "HTTPS Connection Handler" \
     --set correlation-id-response-header:X-Request-Id
  ```

- To designate the names of one or more HTTP request headers that contain an existing correlation ID value, use the `set-connection-handler-prop` option with **`dsconfig`**.

> ⓘ **Note:**
>
> This enables the server to integrate with a larger system consisting of every servers using correlation IDs.
>
> By default, the server generates a new, unique correlation ID for each HTTP request and ignores any correlation ID that might be set on the request.

```
$ dsconfig set-connection-handler-prop --handler-name "HTTPS Connection
 Handler" \
  --set correlation-id-request-header:X-Request-Id \
  --set correlation-id-request-header:X-Correlation-Id \
  --set correlation-id-request-header:Correlation-Id \
  --set correlation-id-request-header:X-Amzn-Trace-Id
```

HTTP correlation ID example

In this example, a request to the Directory REST API is made and the correlation ID enables finding HTTP-specific log messages with LDAP-specific log messages. The response to the API call includes a Correlation-Id header with the value **`a54aee33-c6c6-4467-be25-efd1db7a8b76`**.

```
GET /directory/v1/me?includeAttributes=mail HTTP/1.1
 Accept: */*
 Accept-Encoding: gzip, deflate
 Authorization: Bearer ...
 Connection: keep-alive
 Host: localhost:1443
 User-Agent: HTTPie/0.9.9

 HTTP/1.1 200 OK
 Content-Length: 266
 Content-Type: application/hal+json
 Correlation-Id: ee919049-6710-4594-9c66-28b4ada4b127
 Date: Fri, 02 Nov 2018 15:16:50 GMT
 Request-Id: 369

 {
      "_dn": "uid=user.86,ou=People,dc=example,dc=com",
      "_links": {
         "schemas": [
            {
              "href": "https://localhost:1443/directory/v1/schemas/
inetOrgPerson"
            }
         ],
         "self": {
              "href": "https://localhost:1443/directory/v1/
uid=user.86,ou=People,dc=example,dc=com"
         }
     },
     "mail": [
         "user.86@example.com"
     ]
 }
```

This correlation ID can be used to search the HTTP trace log for matching log records, as follows:

```
$  grep 'correlationID="ee919049-6710-4594-9c66-28b4ada4b127"'
 PingDirectory/logs/debug-trace
      [02/Nov/2018:10:16:50.294 -0500] HTTP REQUEST requestID=369
 correlationID="ee919049-6710-4594-9c66-28b4ada4b127"
 product="PingDirectoryProxy Directory Server" instanceName="ds1"
 startupID="W9ikqA==" threadID=52358 from=[0:0:0:0:0:0:0:1]:58918 method=GET
 url="https://0:0:0:0:0:0:0:1:1443/directory/v1/me?includeAttributes=mail"
 [02/Nov/2018:10:16:50.526 -0500] DEBUG ACCESS-TOKEN-VALIDATOR-PROCESSING
 requestID=369 correlationID="ee919049-6710-4594-9c66-28b4ada4b127"
 msg="Identity Mapper with DN 'cn=User ID Identity Mapper,cn=Identity
 Mappers,cn=config' mapped ID 'user.86' to entry DN
 'uid=user.86,ou=people,dc=example,dc=com'"
 [02/Nov/2018:10:16:50.526 -0500] DEBUG ACCESS-TOKEN-VALIDATOR-PROCESSING
 requestID=369 correlationID="ee919049-6710-4594-9c66-28b4ada4b127"
 accessTokenId="201811020831" msg="Token Validator 'Mock Access Token
 Validator' validated access token with active = 'true', sub = 'user.86',
 owner = 'uid=user.86,ou=people,dc=example,dc=com', clientId = 'client1',
 scopes = 'ds', expiration = 'none', not-used-before = 'none', current time
 = 'Nov 2, 2018 10:16:50 AM CDT' "
 [02/Nov/2018:10:16:50.531 -0500] HTTP RESPONSE requestID=369
 correlationID="ee919049-6710-4594-9c66-28b4ada4b127"
 accessTokenId="201811020831" product="PingDirectoryProxy Directory Server"
 instanceName="ds1" startupID="W9ikqA==" threadID=52358 statusCode=200
 etime=236.932 responseContentLength=266
      [02/Nov/2018:10:16:50.531 -0500] DEBUG HTTP-FULL-REQUEST-AND-RESPONSE
 requestID=369 correlationID="ee919049-6710-4594-9c66-28b4ada4b127"
 accessTokenId="201811020831" product="PingDirectoryProxy Directory
 Server" instanceName="ds1" startupID="W9ikqA==" threadID=52358
 from=[0:0:0:0:0:0:0:1]:58918 method=GET url="https://0:0:0:0:0:0:0:1:1443/
directory/v1/me?includeAttributes=mail" statusCode=200 etime=236.932
 responseContentLength=266 msg="
```

The LDAP log messages associated with this request can also be located:

```
$  grep 'correlationID="ee919049-6710-4594-9c66-28b4ada4b127"'
 PingDirectory/logs/access
 [02/Nov/2018:10:16:50.529 -0500] SEARCH RESULT instanceName="ds1"
 threadID=52358 conn=-371045 op=1657393 msgID=1657394
 origin="Directory REST API" httpRequestID="369"
 correlationID="ee919049-6710-4594-9c66-28b4ada4b127"
 authDN="uid=user.86,ou=people,dc=example,dc=com" requesterIP="internal"
 requesterDN="uid=user.86,ou=People,dc=example,dc=com"
 requestControls="1.3.6.1.4.1.30221.2.5.2" via="app='PingDirectory-
ds1' clientIP='0:0:0:0:0:0:0:1' sessionID='201811020831'
 requestID='ee919049-6710-4594-9c66-28b4ada4b127'"
 base="uid=user.86,ou=people,dc=example,dc=com" scope=0 filter="(&)"
 attrs="mail,objectClass" resultCode=0 resultCodeName="Success" etime=0.684
 entriesReturned=1
 [02/Nov/2018:10:16:50.530 -0500] EXTENDED RESULT
 instanceName="ds1" threadID=52358 conn=-371046 op=1657394
 msgID=1657395 origin="Directory REST API" httpRequestID="369"
 correlationID="ee919049-6710-4594-9c66-28b4ada4b127"
 authDN="cn=Internal Client,cn=Internal,cn=Root DNs,cn=config"
 requesterIP="internal" requesterDN="cn=Internal Client,cn=Internal,cn=Root
 DNs,cn=config" requestControls="1.3.6.1.4.1.30221.2.5.2"
 via="app='PingDirectory-ds1' clientIP='0:0:0:0:0:0:0:1'
 sessionID='201811020831' requestID='ee919049-6710-4594-9c66-28b4ada4b127'"
 requestOID="1.3.6.1.4.1.30221.1.6.1" requestType="Password
 Policy State" resultCode=0 resultCodeName="Success"
 etime=0.542 usedPrivileges="bypass-acl,password-reset"
```

```
 responseOID="1.3.6.1.4.1.30221.1.6.1" responseType="Password Policy State"
 dn="uid=user.86,ou=People,dc=example,dc=com"
 [02/Nov/2018:10:16:50.530 -0500] SEARCH RESULT instanceName="ds1"
 threadID=52358 conn=-371048 op=1657397 msgID=1657398
 origin="Directory REST API" httpRequestID="369"
 correlationID="ee919049-6710-4594-9c66-28b4ada4b127" authDN="cn=Internal
 Client,cn=Internal,cn=Root DNs,cn=config" requesterIP="internal"
 requesterDN="cn=Internal Client,cn=Internal,cn=Root DNs,cn=config"
 requestControls="1.3.6.1.4.1.30221.2.5.2" via="app='PingDirectory-
ds1' clientIP='0:0:0:0:0:0:0:1' sessionID='201811020831'
 requestID='ee919049-6710-4594-9c66-28b4ada4b127'" base="cn=Default Password
 Policy,cn=Password Policies,cn=config" scope=0 filter="(&)" attrs="ds-
cfg-password-attribute" resultCode=0 resultCodeName="Success" etime=0.065
 preAuthZUsedPrivileges="bypass-acl,config-read" entriesReturned=1
```

Configuring proxy transformations

The PingDirectoryProxy Server provides proxy transformations to alter the contents of client requests as they are sent from the client to the LDAP external server. Proxy transformations can also be used to alter the responses sent back from the server to the client, including altering or omitting search result entries. The Directory Proxy Server provides the following types of data transformations:

- **Attribute mapping**. This transformation rewrites client requests so that references to one attribute type may be replaced with an alternate attribute type. The Directory Proxy Server can perform extensive replacements, including attribute names used in DNs and attribute names encoded in the values of a number of different controls and extended operations. For example, a client requests a `userid` attribute, which is replaced with `uid` before being forwarded on to the backend server. This mapping applies in reverse for the response returned to the client.
- **Default value**. This transformation instructs the Directory Proxy Server to include a static attribute value in search results being sent back to the client, in ADD requests being forwarded to an external server, or both. For example, a value of "marketing" for `businessCategory` could be returned for all search results under the base DN `ou=marketing,dc=example,dc=com`.
- **DN mapping**. This transformation rewrites client requests so that references to entries below a specified DN will be mapped to appear below another DN. For example, references to entries below `o=example.com` could be rewritten so that they are below `dc=example,dc=com` instead. The mapping applies in reverse for the response returned to the client.
- **Groovy scripted**. This custom transformation is written in Groovy and does not need to be compiled, though they use the Server SDK. These scripts make it possible to alter requests and responses in ways not available using the transformations provided with the Directory Proxy Server.
- **Suppress attribute**. This proxy transformation allows you to exclude a specified attribute from search result entries. It also provides the ability to reject add, compare, modify, modify DN, or search requests if they attempt to reference the target attribute.
- **Suppress entry**. This proxy transformation allows you to exclude any entries that match a specified filter from a set of search results. Search requests are transformed so that the original filter will be ANDed with a NOT filter containing the exclude filter. For example, if the suppression filter is "`(objectClass=secretEntry)`", then a search request with a filter of "`(uid=john.doe)`" will be transformed so that it has a filter of "`(&(uid=john.doe)(!(objectClass=secretEntry)))`".
- **Simple to external bind**. This proxy transformation may be used to intercept a simple bind request and instead process the bind as a SASL EXTERNAL bind. If the SASL EXTERNAL bind fails, then the original simple bind request may or may not be processed, depending on how you configure the server.
- **Third-party scripted**. This custom transformation is created using the Server SDK, making it possible to alter requests and responses in ways not available using the transformations provided with the Directory Proxy Server.

Configuring proxy transformations using dsconfig

Steps

1. Use the **`dsconfig`** tool to create and configure a proxy transformation.

```
$ bin/dsconfig
```

2. Enter the connection parameters (for example, host name, port, bind DN and bind DN password).
3. In the Directory Proxy Server main menu, enter the number associated with proxy transformations. On the Proxy Transformation menu, enter the number to create a new proxy transformation.
4. Select the type of proxy transformation you want to create. In this example, we create an attribute mapping transformation. Then, enter a name for the new transformation.

```
 >>>> Enter a name for the Attribute Mapping Proxy Transformation that you
want to create: userid-to-uid
```

5. Indicate whether you want the transformation to be enabled by default.

```
Select a value for the 'enabled' property:

  1) true
  2) false

  ?) help
  c) cancel
  q) quit

Enter choice [c]: 1
```

6. Specify the name of the client attribute that you want to remap to a target attribute. Note that this attribute must not be equal to the target attribute.

```
Enter a value for the 'source-attribute' property: userid
```

7. Specify the name of the target attribute to which the client attribute should be mapped.

```
Enter a value for the 'target-attribute' property: uid
```

8. The properties of your new proxy transformation are displayed. If you want to make any further modifications, enter the number corresponding to the property. Enter `f` to finish the creation of the proxy transformation.

```
Enter choice [b]: f
```

The transformation now needs to be assigned to a request processor. To create an initial request processor, see the next section.

Configuring request processors

A request processor is responsible for handling client requests by passing the request through a load-balancing algorithm or one or more subordinate request processors. The request processor is also the Directory Proxy Server component that performs proxy transformations. You can create one of the following types of request processors:

- **Proxying request processor**. This request processor is responsible for passing allowed operations through a load balancing algorithm. Proxy transformations can be applied to requests and responses that are processed. Multiple servers may be configured to provide high availability and load balancing, and various transformations may be applied to the requests and responses that are processed.
- **Entry-balancing request processor**. This request processor is used to distribute entries under a common parent entry among multiple backend sets. A backend set is a collection of replicated directory servers that contain identical portions of the data. This request processor uses multiple, subordinate

proxying request processors to process operations and maintains in-memory indexes to speed the processing of specific search and modify operations.

- **Failover request processor**. This request processor performs ordered failover between subordinate proxying processors, sometimes with different behavior for different types of operations.

Configuring request processors using dsconfig

Steps

1. Use the **`dsconfig`** tool to create and configure a request processor.

```
$ bin/dsconfig
```

2. Specify the host name, connection method, port number, and bind DN as described in previous procedures.
3. In the Directory Proxy Server main menu, enter the number associated with Request Processor configuration and select the option to create a new Request Processor.
4. Select an existing request processor to use as a template for creating a new one or enter `n` to create one from scratch. In this example, we create a new proxying request processor from scratch. You will be required to choose an existing load balancing algorithm or create a new one to complete the create of the request processor. Below is the configuration of the proxying request processor after selection of the load balancing algorithm.

```
Property                      Value(s)
-------------------------------------------------------------
1)  description               -
2)  enabled                   true
3)  allowed-operation         abandon, add, bind, compare, delete,
                              extended, modify, modify-dn, search
4)  load-balancing-algorithm dc_example_dc_com-fewest-operations
5)  transformation            -
6)  referral-behavior         pass-through
7)  supported-control         account-usable, assertion,
                              authorization-identity,
                              get-authorization-entry,
                              get-effective-rights, get-server-id,
                              ignore-no-user-modification,
                              intermediate-client, manage-dsa-it,
                              matched-values, no-op, password-policy,
                              permissive-modify, post-read, pre-read,
                              proxied-authorization-v1,
                              proxied-authorization-v2,
                              real-attributes-only, retain-identity,
                              subentries, subtree-delete,
                              virtual-attributes-only
8)  supported-control-oid     -

?)  help
f)  finish - create the new Proxying Request Processor
d)  display the equivalent dsconfig arguments to create this object
b)  back
q)  quit
```

5. Review the configuration properties of the new request processor. If you are satisfied, enter `f` to finish. For the request processor to be used, it must be associated with a subtree view.

Passing LDAP controls with the proxying request processor

About this task

If your deployment does not use entry balancing and requires the use of LDAP controls not defined in the request processor's supported-control property, configure the Directory Proxy Server to forward these controls correctly. This is done by configuring the **`supported-control-oid`** property to define the request OID of the LDAP control. The Directory Proxy Server updates the root DSE **`supportedControl`** attribute with the values entered for the **`supported-control-oid`** property.

Configuring server affinity

About this task

The Directory Proxy Server supports the ability to forward a sequence of requests to the same external server if specific conditions are met. This feature, called server affinity, is applied by the load balancing algorithms. The following server affinity methods are available in the Directory Proxy Server:

- **Client Connection**. Requests from the same Directory Proxy Server client connection are consistently routed to the same external server.
- **Client IP**. Directory Proxy Server client requests coming from the same client IP address are routed to the same external server.
- **Bind DN**. Requests from all client connections authenticated as the same bind DN are routed to the same external server.

For each algorithm, you can specify the set of operations for which an affinity will be established, as well as the set of operations for which affinity will be used. Affinity assignments have a time-out value so that they are in effect for some period of time after the last operation that may cause the affinity to be set or updated.

In this example, we create a bind DN server affinity provider for any client requesting write operations to have subsequent requests, whether read or write, forwarded to the same external server. The affinity period will last for 30 seconds after the last write request.

Steps

1. Use the **`dsconfig`** tool to configure server affinity. Specify the host name, connection method, port number, and bind DN as described in previous procedures.

```
$ bin/dsconfig
```

2. In the Directory Proxy Server main menu, enter the number associated with server affinity provider configuration
3. On the Server Affinity menu, enter the number corresponding to creating a new server affinity provider.
4. Enter a name for your new server affinity provider.

```
>>>> Enter a name for the Bind DN Server Affinity Provider
that you want to create: Affinity for Writing Applications
```

5. Indicate whether you want the server affinity provider to be enabled for use by the Directory Proxy Server. In this example, enter `1` to enable to the server affinity provider.
6. Next, configure the properties of the server affinity provider. For example, you can customize the types of operations for which affinity may be set and the types of operations for which affinity may be used, as well as the length of time for which the affinity should persist. This example illustrates the properties of the bind DN server affinity provider.

```
>>>> >>>> Configure the properties of the Bind DN Server Affinity Provider

        Property                  Value(s)
        ----------------------------------------------------------
    1)  description               -
```

```
    2)  enable                      true
    3)  affinity-duration           30 s
    4)  set-affinity-operation      add, delete, modify, modify-dn
    5)  use-affinity-operation      add, bind, compare, delete, modify,
                                    modify-dn, search

    ?)  help
    f)  finish - create the new Bind DN Server Affinity Provider
    d)  display the equivalent dsconfig arguments to create this object
    b)  back
    q)  quit

Enter choice [b]:
```

7. Review the properties of the server affinity provider. If you are satisfied, enter `f` to finish. Once defined, the affinity provider can now be assigned to a load balancing algorithm.

Configuring subtree views

About this task

You provide clients access to a specific portion of the DIT creating a subtree view and assigning it to a client connection policy. You can configure subtree views from the command line or using the Administrative Console.

When you create a subtree view, you provide the following information to configure its properties:

- Subtree view name
- Base DN managed by the subtree view
- Request processor used by the subtree view to route requests. If one does not exist already, you will create a new one.

Steps

1. Use **`dsconfig`** to configure a subtree view.
2. In the Directory Proxy Server main menu, enter the number associated with subtree view configuration
3. In the Subtree View menu, enter the number corresponding to creating a new subtree view.
4. Enter a name for the subtree view.
5. Enter the base DN of the subtree managed by this subtree view.

```
Enter a value for the 'base-dn' property:dc=example,dc=com
```

6. Select a request processor for this subtree view to route requests or make the appropriate selection to create a new one.

```
Select a Request Processor for the 'request-processor' property:

1) dc_example_dc_com-req-processor
2) Create a new Request Processor

?) help
c) cancel
q) quit

Enter choice [c]: 1
```

7. Review the properties of the subtree view. If you are satisfied, enter `f` to finish.

```
>>>> Configure the properties of the Subtree View
 >>>> via creating 'example.com' Subtree View

        Property           Value(s)
```

```
        ------------------------------------------------
    1)  description         -
    2)  base-dn             "dc=example,dc=com"
    3)  request-processor   dc_example_dc_com-req-processor

    ?)  help
    f)  finish - create the new Subtree View
    d)  display the equivalent dsconfig arguments to create
        this object
    b)  back
    q)  quit
```

Once configured, you can assign one or more subtree views to any client connection policies.

Client connection policy configuration

Client connection policies help distinguish what portions of the DIT the client can access.

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

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

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

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

About the client connection policy

Client connection policies are based on two factors.

Connection criteria

The connection criteria are used in many areas within the server. They are used by the client connection policies, but they can be used in other instances when the server needs to perform matching based on connection-level properties, such as filtered logging. A single connection can match multiple connection criteria definitions.

Evaluation order index

If multiple client connection policies are defined in the server, then each of them must have a unique value for the `evaluation-order-index` property. The client connection policies are evaluated in order of ascending evaluation order index. If a client connection does not match the criteria for any defined client connection policy, then that connection will be terminated.

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

When a client connection policy is assigned
A client connection policy can be associated with a client connection at the following times.

- When the connection is initially established. This association occurs exactly once for each client connection.
- After completing processing for a StartTLS operation. This association occurs once at most for a client connection because StartTLS cannot be used more than once on a particular connection. You can not stop using StartTLS while keeping the connection active.
- After completing processing for a bind operation. This association occurs zero or more times for a client connection because the bind request can be processed many times on a given connection.

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

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

Restricting the type of search filter used by clients
You can restrict the types of search filters that a given client might be allowed to use to prevent the use of potentially expensive filters, like range or substring searches.

You can use the `allowed-filter-type` property to provide a list of filter types that can be included in the search requests from clients associated with the client connection policy. This setting is only used if search is included in the set of allowed operation types. This restriction only applies to searches with a scope other than `baseObject`, such as searches with a scope of `singleLevel`, `wholeSubtree`, or `subordinateSubtree`.

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

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

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

Defining Request Criteria

The client connection policy provides several properties that allow you to define the kinds of requests that it can issue. The `required-operation-request-criteria` property causes the server to reject any requests that do not match the referenced request criteria. The `prohibited-operation-request-criteria` property causes the server to reject any request that matches the referenced request criteria.

Setting Resource Limits

A client connection policy can specify resource limits, helping to ensure that no single client monopolizes server resources. The resource limits are applied in addition to any global configuration resource limits. In other words, a client connection policy cannot grant additional resources beyond what is set in the global configuration. If a client connection exceeds either a globally-defined limit or a policy limit, then it is terminated.

ⓘ **Note:** The Directory Proxy Server's global configuration can enforce limits on the number of concurrent connections that can be established in the following ways:

- Limit the total number of concurrent connections to the server.
- Limit the total number of concurrent connections from the same IP address.
- Limit the total number of concurrent connections authenticated as the same bind DN.

Defining the operation rate

Configure the maximum operation rate for individual client connections and collectively for all connections associated with a client connection policy.

If the operation rate limit is exceeded, the Directory Proxy Server can either reject the operation or terminate the connection. You can define multiple rate limit values, making it possible to fine tune limits for both a long term average operation rate and short term operation bursts. For example, you can define a limit of one thousand operations per second and one million operations per day, which works out to an average of fewer than twelve operations per second, but with bursts of up to one thousand operations per second.

Specify rate limit strings as a maximum count, followed by a slash and a duration. The count portion must contain an integer and can be followed by the following multipliers:

- k (to indicate that the integer should be interpreted as thousands)
- m (to indicate that the integer should be interpreted as millions)
- g (to indicate that the integer should be interpreted as billions)

The duration portion must contain a time unit of milliseconds (ms), seconds (s), minutes (m), hours (h), days (d), or weeks (w) and can be preceded by an integer to specify a quantity for that unit.

For example, the following are valid rate limit strings:

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

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

Client connection policy deployment example

In this example scenario, we assume the following:

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

For more information on this client connection policy deployment example, see the following topics:

- *Define the connection policies* on page 396
- *How the policy is evaluated* on page 396
- *Configuring a client connection policy using the console* on page 397
- *Configuring a client connection policy using dsconfig* on page 398
- *Restricting server access based on client IP address* on page 399

Define the connection policies
Set a per-client connection policy limit on the number of connections that can be associated with a particular client connection policy

Define at least two client connection policies, one for each of the two clients. Each policy must have different connection criteria for selecting the policy with which a given client connection should be associated.

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

Define the following three client connection policies:

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

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

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

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

How the policy is evaluated
Whenever a connection is established, the server associates the connection with exactly one client connection policy.

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

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

After the client performs a bind operation to authenticate to the server, then the client connection policies are re-evaluated. If Client 2 performs the bind, then the Client 1 Connection Policy does not match, but the Client 2 Connection Policy does, so the connection is re-associated with that client connection policy. Whenever a connection is associated with a client connection policy, the server checks to see if the

maximum number of client connections have already been associated with that policy. If so, then the newly-associated connection is terminated.

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

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

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

Configuring a client connection policy using dsconfig

Steps

1. Use the **`dsconfig`** tool to create and configure a client connection policy.

```
$ bin/dsconfig
```

2. Enter the connection parameters to the server (for example, host name, connection method, port, bind DN and bind DN password).
3. In the Directory Proxy Server main menu, enter the number associated with client connection policy configuration. Then enter the number to create a new client connection policy.

```
>>>> Client connection policy menu

What would you like to do?

  1) List existing client connection policies
  2) Create a new client connection policy
  3) View and edit an existing client connection policy
  4) Delete an existing client connection policy

  b) back
  q) quit

Enter choice [b]: 2
```

4. Enter `n` to create a new client connection policy from scratch.

```
>>>> Select an existing Client Connection Policy to use as a
template for the new Client Connection Policy configuration or
'n' to create one from scratch:

  1) default

  n) new Client Connection Policy created from scratch
```

```
  c) cancel
  q) quit
```

5. Enter a name for the new client connection policy.

```
Enter the 'policy-id' for the Client Connection Policy that you

want to create: new_policy
```

6. Indicate whether you want the policy to be enabled by default.

```
Select a value for the 'enabled' property:

  1) true
  2) false

  ?) help
  c) cancel
  q) quit

Enter choice [c]: 1
```

7. Provide a value for the `evaluation-order-index` property. Client connection policies with a lower index will be evaluated before those with a higher index.

```
Enter a value for the 'evaluation-order-index' property: 2
```

8. The properties of your new client connection policy are displayed. If you want to make any further modifications, enter the number corresponding to the property. Enter `f` to finish the creation of the client connection policy.

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

Configuring Globally Unique attributes

The PingDirectoryProxy Server supports a Globally Unique Attributes feature that ensures uniqueness for any value defined for a set of attributes within a subtree view. You can also configure when the server checks for attribute conflicts, either prior to any add, modify, or modify DN change request (pre-commit) or after the successful completion of a change request (post-commit).

About the Globally Unique Attribute plugin

The Directory Proxy Server supports a Globally Unique Attribute plugin that prevents any value within a defined set of attributes to appear more than once in any entry for one or more subtree views. Administrators can also configure whether conflict validation should be checked before an add, modify, or modify DN request to one or more backend servers or after the change has successfully completed.

For example, if the `pre-commit-validation` property is enabled, the Globally Unique Attribute Plugin performs one or more searches to determine whether any entries conflict with the change (i.e., add, modify, or modify DN). If a conflict is detected, then the change request will be rejected. If the `post-commit-validation` property is enabled, after the change has been processed, the server performs one or more searches to determine if a conflict was created in multiple servers at the same time. If a conflict is detected in this manner, then an administrative alert will be generated to notify administrators of the problem so that they can take any manual corrective action.

ⓘ **Note:** The Globally Unique Attribute plugin will attempt to detect and/or prevent unique attribute conflicts for changes processed through this Directory Proxy Server, but it cannot detect conflicts introduced by changes applied by clients communicating *directly* with backend servers.

We recommend that the Unique Attribute plugin be enabled for all backend servers with the same configuration, so that conflicts can be detected within individual backend server instances. However, the Unique Attribute plugin alone may not be sufficient for cases in which the content is split across multiple

sets of servers (e.g., in an entry-balanced environment or in proxy configurations with different branches on different servers).

The LDAP SDK uniqueness request control can be used for enforcing uniqueness on a per-request basis. See the LDAP SDK documentation and the **`com.unboundid.ldap.sdk.unboundidds.controls.UniquenessResponseControl`** class for using the control. See the ASN.1 specification to implement support for it in other APIs.

In general, note the following points about pre-commit validation versus post-commit validation:

- Pre-commit validation is the only mechanism that can try to prevent conflicts. It will increase the processing time for add, modify, and modify DN operations because the necessary searches to look for conflicts happen before the update request is forwarded to any backend servers.
- Post-commit validation will only let you know (via administrative alert) about conflicts that already exist in the data. It can't prevent conflicts, but can allow you to deal with them in a timely manner. It also operates during the post-response phase, so it won't affect the processing time for the associated write operation.
- In most cases, pre-commit validation should be sufficient to prevent conflicts, although we recommend that you periodically run the **`identify-unique-attribute-conflicts`** tool to find any conflicts that may have arisen. If you want to mitigate any risks due to conflicts being generated by concurrent operations in different servers, then using both **`pre-commit-validation`** and **`post-commit-validation`** properties provides the best combination of preventing most conflicts in advance, and detecting and alerting about conflicts that arise from concurrent writes.

For more detailed information about the plugin, see the *Directory Proxy Server Reference (HTML)*

Configuring the Globally Unique Attribute plugin

About this task

The following example shows how to configure the Globally Unique Attribute plugin. The example defines an attribute set consisting of the `telephoneNumber` and `mobile` attributes within the "test-view" subtree view. The `multiple-attribute-behavior` property determines the scope of how attributes may differ among entries and is the same property for the Directory Server plugin. The property is set to `unique-across-all-attributes-including-in-same-entry`, which indicates that the `telephone` and `mobile` attributes must be unique throughout the subtree view, even within an entry. The `pre-commit-validation` property ensures that the Globally Unique Attribute Plugin performs one or more searches to determine whether any entries conflict with the change (i.e., add, modify, or modify DN). If a conflict is detected, then the change request will be rejected.

Note that all configured attributes should be indexed for equality in all backend servers.

Steps

- Run **`dsconfig`** to create the Globally Unique Attribute plugin. The server checks that any add, modify, or modify DN request does not conflict with any attribute values in the entries. If a conflict exists, the change request is rejected.

```
$ bin/dsconfig create-plugin \
  --plugin-name "Globally-Unique telephone and mobile" \
  --type globally-unique-attribute \
  --set enabled:true \
  --set type:telephoneNumber \
  --set type:mobile \
  --set subtree-view:test-view \
  --set multiple-attribute-behavior:unique-across-all-attributes-including-
in-same-entry \
  --set pre-commit-validation:all-available-backend-servers
```

Configuring the Global Referential Integrity plugin

The PingDirectoryProxy Server supports a global referential integrity plugin mechanism that maintains DN references from a specified set of attributes to entries that exist in the server (e.g., between the members values of a static group and the corresponding user entries). The plugin intercepts delete and modify DN operations and updates any references to the target entry. For a delete operation, any references to the target entry are removed. For modify DN operations, any references to the target entry are updated to reflect the new DN of the entry.

The plugin is similar to the Directory Server Referential Integrity plugin but does not have an asynchronous mode. When enabled on the Directory Proxy Server, the client response will be delayed until the referential integrity processing is complete. For Directory Proxy Server deployments not using entry balancing and using Directory Server external servers, it is best to instead use the Referential Integrity plugin on the Directory Server.

An equality index must be defined on all attributes referenced within the Global Referential Integrity plugin across all external servers.

Sample Global Referential Integrity plugin

Steps

- Use **`dsconfig`** to configure the Global Referential Integrity plugin. The plugin ensures that the `member`, `uniqueMember`, and `manager` attributes maintain their DN references in the defined subtree views. Note that any attributes for which referential integrity should be maintained should have values which are DNs and should be indexed for equality in all backend servers.

```
$ bin/dsconfig create-plugin \
  --plugin-name "Global Referential Integrity" \
  --type global-referential-integrity \
  --set "enabled:true" \
  --set "attribute-type:member" \
  --set "attribute-type:uniqueMember" \
  --set "attribute-type:manager" \
  --set "subtree-view:employee-view" \
  --set "subtree-view:groups-view"
```

Configuring an Active Directory Server back-end

Configuring an Active Directory server back-end requires a **`dsconfig`** script. The following settings are required for an Active Directory server:

- **`verify-credentials-method:bind-on-existing-connections`**, and **`authorization-method:rebind`**

  Active Directory does not support **`proxy-as`**. Existing connections must be reused.
- **`set max-connection-age:5m`**, and **`health-check-pooled-connections:true`**

  Active Directory drops idle connections after 15 minutes. The proxy needs to refresh the connection pool in a shorter interval.

The following example **`dsconfig`** script configures two Active Directory servers (AD-SRV1 and AD-SRV2).

```
dsconfig set-ldap-health-check-prop --check-name "Consume Admin Alerts" \
  --reset use-for-all-servers

dsconfig set-trust-manager-provider-prop \
  --provider-name "Blind Trust" \
  --set enabled:true
```

```
dsconfig create-external-server --server-name AD-SRV1 --type active-directory
 \
  --set server-host-name:example.server \
  --set server-port:636 \
  --set bind-dn:cn=ProxyUser,dc=dom-ad2,dc=local \
  --set password:password --set connection-security:ssl \
  --set key-manager-provider:Null --set trust-manager-provider:"Blind Trust" \
  --set authorization-method:rebind \
  --set verify-credentials-method:bind-on-existing-connections \
  --set max-connection-age:5m \
  --set health-check-pooled-connections:true

dsconfig create-external-server --server-name AD-SRV2 --type active-directory
 \
  --set server-host-name:example.server \
  --set server-port:636 \
  --set bind-dn:cn=ProxyUser,dc=dom-ad2,dc=local \
  --set password:password \
  --set connection-security:ssl \
  --set key-manager-provider:Null \
  --set trust-manager-provider:"Blind Trust" \
  --set authorization-method:rebind \
  --set verify-credentials-method:bind-on-existing-connections \
  --set max-connection-age:5m \
  --set health-check-pooled-connections:true

dsconfig create-load-balancing-algorithm --algorithm-name AD-LBA \
  --type fewest-operations \
  --set enabled:true \
  --set backend-server:AD-SRV1 \
  --set backend-server:AD-SRV2 \
  --set use-location:false

dsconfig create-request-processor --processor-name AD-Proxy --type proxying \
  --set load-balancing-algorithm:AD-LBA

dsconfig create-subtree-view --view-name AD-View \
  --set base-dn:dc=dom-ad2,dc=local \
  --set request-processor:AD-Proxy

dsconfig set-client-connection-policy-prop --policy-name default \
  --set subtree-view:AD-View
```

Configuring PingOne to use SSO for the PingData Administrative Console

Allow administrative users to single sign-on (SSO) to the PingData admin console from PingOne.

Before you begin

You need the following to complete this process:

- A configured PingData server. This server will host the the PingOne administration console console that is being configured for SSO.
- A PingOne for Customers account. For more information, see *Getting started with PingOne for Customers*.
- Access to the the PingOne administration console console. For more information, see *Using the beta version of My Ping*.

Steps

1. In the PingOne administration console, add a link to the PingOne solutions home page.
   a. In the the PingOne administration console admin console, click **Add Environment**.

      If you're adding PingDirectory Server or PingDataGovernance Server to an existing environment, click the name of an environment, click the **Plus** icon and click **Add** to add PingDirectory.
   b. To create an environment, on the **Create Environment** page, select from **Customers**, **Workforce**, or **Custom**.
   c. Select **PingDirectory** and **PingOne for Customers**.
   d. Click **Next**.
   e. Select `It's already been deployed`.
   f. In the **Enter Admin URL** field, enter `https://<hostname>:<port>/console/login`, replacing the bracketed variables with the PingData server's hostname and HTTP port.
   g. Click **Next**.
   h. In the **Environment Name** field, enter a name for this environment.
   i. Optional: In the **Description** field, enter a description for the environment.
   j. From the **Region** list, select your data center region.
   k. From the **License** list, select the license for this environment.
   l. Click **Finish**.
2. To configure the matching administrator accounts for PingOne and the PingData server, go to the PingOne dashboard for the environment that will be used with the PingData server and repeat the following steps for each PingOne user for whom you want to enable SSO.
   a. In the PingOne administration console, on your environment line, click the **PingOne** icon.
   b. In PingOne, go to **Identities**.
   c. On the line of the administrative user you want to configure, click the **Expand** icon.
   d. Run the following **`dsconfig`** command against the PingData server, replacing the bracketed fields with the values of the administrative user.

      ```
      dsconfig create-root-dn-user --user-name <Username> \
        --set first-name:<Given Name> \
        --set last-name:<Family Name>
      ```
3. Register the Administrative Console with PingOne.
   a. Go to *Add an application - Web application* and follow the instructions in the **Add an OIDC application** subsection.
   b. Enter the application properties as shown in the following table.

Property	Value
Application Name	PingData Administrative Console
Description	Application for the PingData Administrative Console
Redirect URLs	`https://<hostname>:<port>/console/oidc/cb`
Attribute Mapping	'Username' = 'sub'

> ⓘ **Note:**
>
> Fill in the bracketed values in **Redirect URLs** with your PingData server's hostname and HTTP port.

4. Edit the listed properties for the newly created application so that the properties have the values show in the following table, following the instructions in *Edit an application - OIDC* in the PingOne Administration Guide.

Property	Value
Response Type	Code
Grant Type	Authorization Code
Token Endpoint Authentication Method	Client Secret Basic

5. Record the values for the following application properties to use in later steps:
   - Issuer
   - Client ID
   - Client Secret
6. Create a copy of the `PingDirectory/config/sample-dsconfig-batch-files/enable-pingone-admin-console-sso.dsconfig` file, leaving the source file as-is.
7. Open the copy of the file and replace the bracketed values with the values from step 5.
8. Run the file using the following command.

```
dsconfig --batch-file \
    enable-pingone-admin-console-sso-copy.dsconfig \
    --no-prompt
```

9. Click the link to the PingData server from the PingOne solutions home page.
   A PingOne sign on page displays.
10. Sign on using the administrative user credentials.
    The Administrative console index page displays.

Managing Access Control

The PingDirectoryProxy Server provides a fine-grained access control model to ensure that users are able to access the information they need, but are prevented from accessing information that they should not be allowed to see. It also includes a privilege subsystem that provides even greater flexibility and protection in many key areas.

This chapter presents the access control model and how it applies to the Directory Proxy Server.

Overview of access control

The access control model uses access control instructions (ACIs), which are stored in the `aci` operational attribute, to determine what a user or a group of users can do with a set of entries, down to the attribute level. The operational attribute can appear on any entry and affects the entry or any subentries within that branch of the directory information tree (DIT).

Access control instructions specifies four items:

- **Resources**. *Resources* are the targeted items or objects that specifies the set of entries and/or operations to which the access control instruction applies. For example, you can specify access to certain attributes, such as the `cn` or `userPassword` password.
- **Name**. *Name* is the descriptive label for each access control instruction. Typically, you will have multiple access control instructions for a given branch of your DIT. The access control name helps describe its purpose. For example, you can configure an access control instructions labelled "ACI to grant full access to administrators."

- **Clients**. *Clients* are the users or entities to which you grant or deny access. You can specify individual users or groups of users using an LDAP URL. For example, you can specify a group of administrators using the LDAP URL: `groupdn="ldap:///cn=admins,ou=groups,dc=example,dc=com."`
- **Rights**. *Rights* are permissions granted to users or client applications. You can grant or deny access to certain branches or operations. For example, you can grant `read` or `write` permission to a `telephoneNumber` attribute.

Key access control features

The PingDirectoryProxy Server provides important access control features that provide added security for the Directory Proxy Server's entries.

Improved validation and security

The Directory Proxy Server provides an access control model with strong validation to help ensure that invalid ACIs are not allowed into the server. For example, the Directory Proxy Server ensures that all access control rules added over LDAP are valid and can be fully parsed. Any operation that attempts to store one or more invalid ACIs are rejected. The same validation is applied to ACIs contained in data imported from an LDIF file. Any entry containing a malformed `aci` value will be rejected.

As an additional level of security, the Directory Proxy Server examines and validates all ACIs stored in the data whenever a backend is brought online. If any malformed ACIs are found in the backend, then the server generates an administrative alert to notify administrators of the problem and places itself in lockdown mode. While in lockdown mode, the server only allows requests from users who have the `lockdown-mode` privilege. This action allows administrators to correct the malformed ACI while ensuring that no sensitive data is inadvertently exposed due to an access control instruction not being enforced. When the problem has been corrected, the administrator can use the **`leave-lockdown-mode`** tool or restart the server to allow it to resume normal operation.

Global ACIs

Global ACIs are a set of ACIs that can apply to entries anywhere in the server (although they can also be scoped so that they only apply to a specific set of entries). They work in conjunction with access control rules stored in user data and provide a convenient way to define ACIs that span disparate portions of the DIT.

In the PingDirectoryProxy Server, global ACIs are defined within the server configuration, in the `global-aci` property of configuration object for the access control handler. They can be viewed and managed using configuration tools like **`dsconfig`** and the Administrative Console.

The global ACIs available by default in the PingDirectoryProxy Server include:

- Allow anyone (including unauthenticated users) to access key attributes of the root DSE, including: `namingContexts`, `subschemaSubentry`, `supportedAuthPasswordSchemes`, `supportedControl`, `supportedExtension`, `supportedFeatures`, `supportedLDAPVersion`, `supportedSASLMechanisms`, `vendorName`, and `vendorVersion`.
- Allow anyone (including unauthenticated users) to access key attributes of the subschema subentry, including: `attributeTypes`, `dITContentRules`, `dITStructureRules`, `ldapSyntaxes`, `matchingRules`, `matchingRuleUse`, `nameForms`, and `objectClasses`.
- Allow anyone (including unauthenticated users) to include the following controls in requests made to the server: authorization identity request, manage DSA IT, password policy, real attributes only, and virtual attributes only.
- Allow anyone (including unauthenticated users) to request the following extended operations: get symmetric key, password modify request, password policy state, StartTLS, and Who Am I?

Access controls for public or private backends

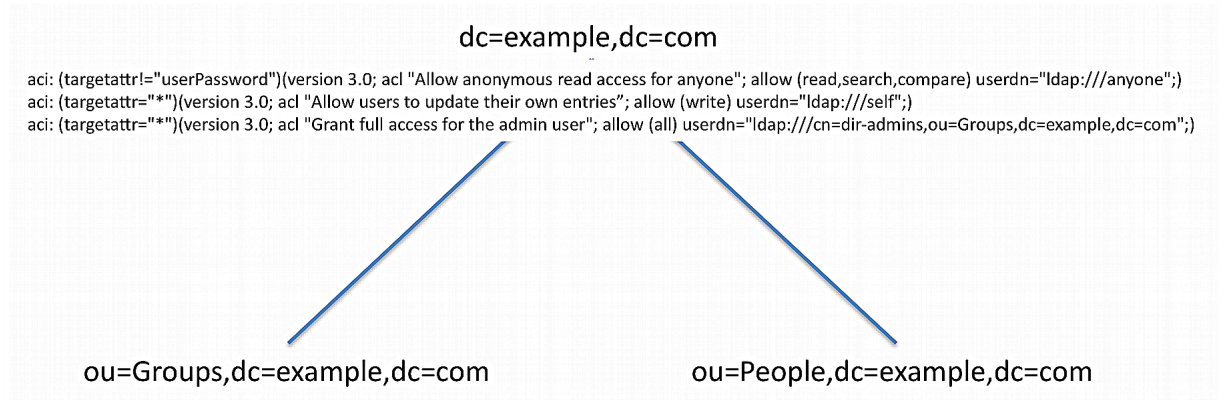
The PingDirectoryProxy Server classifies backends as either public or private, depending on their intended purpose. A private backend is one whose content is generated by the Directory Proxy Server itself (for example, the root DSE, monitor, and backup backends), is used in the operation of the server (for example, the configuration, schema, task, and trust store backends), or whose content is maintained by

the server (for example, the LDAP changelog backend). A public backend is intended to hold user-defined content, such as user accounts, groups, application data, and device data.

The PingDirectoryProxy Server access control model also supports the distinction between public backends and private backends. Many private backends do not allow writes of any kind from clients, and some of the private backends that do allow writes only allow changes to a specific set of attributes. As a result, any access control instruction intended to permit or restrict access to information in private backends should be defined as global ACIs, rather than attempting to add those instructions to the data for that private backend.

General format of the access control rules

Access control instructions (ACIs) are represented as strings that are applied to one or more entries within the Directory Information Tree (DIT). Typically, an ACI is placed on a subtree, such as `dc=example,dc=com`, and applies to that base entry and all entries below it in the tree. The Directory Proxy Server iterates through the DIT to compile the access control rules into an internally-used list of denied and allowed targets and their permissable operations. When a client application, such as **`ldapsearch`**, enters a request, the Directory Proxy Server checks that the user who binds with the server has the necessary access rights to the requested search targets. ACIs are cumulatively applied, so that a user who may have an ACI at an entry, may also have other access rights available if ACIs are defined higher in the DIT and are applicable to the user. In most environments, ACIs are defined at the root of a main branch or a subtree, and not on individual entries unless absolutely required.



MY TITLE ACI

An access control rule has a basic syntax as follows:

```
aci : (targets) (version 3.0; acl "name";
  permissions
  bind rules
  ;)
```

Access Control Components

Access Control Component	Description
targets	Specifies the set of entries and/or attributes to which an access control rule applies. Syntax: *(target keyword = || != expression)*
name	Specifies the name of the ACI.
permissions	Specifies the type of operations to which an access control rule might apply. Syntax: *allow||deny (permission)*

Access Control Component	Description
bind rules	Specifies the criteria that indicate whether an access control rule should apply to a given requestor. Syntax: *bind rule keyword* = ||*!= expression;*. The bind rule syntax requires that it be terminated with a ";".

Summary of access control keywords

This section provides an overview of the keywords supported for use in the PingDirectoryProxy Server access control implementation.

Targets

A target expression specifies the set of entries and/or attributes to which an access control rule applies. The *keyword* specifies the type of target element. The *expression* specifies the items that is targeted by the access control rule. The operator is either the equal ("=") or not-equal ("!="). Note that the "!=" operator cannot be used with `targattrfilters` and `targetscope` keywords. For specific examples on each target keyword, see the section *Working with Targets*.

```
(keyword [=||!=]expression)
```

The following keywords are supported for use in the target portion of ACIs:

Summary of Access Control Target Keywords

Target Keyword	Description	Wildcards
extop	Specifies the OIDs for any extended operations to which the access control rule should apply.	No
target	Specifies the set of entries, identified using LDAP URLs, to which the access control rule applies.	**Yes**
targattrfilters	Identifies specific attribute values based on filters that may be added to or removed from entries to which the access control rule applies.	**Yes**
targetattr	Specifies the set of attributes to which the access control rule should apply.	**Yes**
targetcontrol	Specifies the OIDs for any request controls to which the access control rule should apply.	No
targetfilter	Specifies one or more search filters that may be used to indicate the set of entries to which the access control should apply.	**Yes**
targetscope	Specifies the scope of entries, relative to the defined target entries or the entry containing the ACI fi there is no target, to which the access control rule should apply.	No

Permissions

Permissions indicate the types of operations to which an access control rule might apply. You can specify if the user or group of users are allowed or not allowed to carry out a specific operation. For example, you would grant read access to a targeted entry or entries using "allow (read)" permission. Or you can specifically deny access to the target entries and/or attributes using the "deny (read)" permission. You can list out multiple permissions as required in the ACI.

```

```

```
allow (permission1 ...,
permission2
,...permissionN)
```

```
deny (permission1 ...,
permission2
,...permissionN)
```

The following keywords are supported for use in the permissions portion of ACIs:

Summary of Access Control Permissions

Permission	Description
add	Indicates that the access control should apply to add operations.
compare	Indicates that the access control should apply to compare operations, as well as to search operations with a base-level scope that targets a single entry.
delete	Indicates that the access control should apply to delete operations.
export	Indicates that the access control should only apply to modify DN operations in which an entry is moved below a different parent by specifying a new superior DN in the modify DN request. The requestor must have the `export` permission for operations against the entry's original DN. The requestor must have the `import` permission for operations against the entry's new superior DN. For modify DN operations that merely alter the RDN of an entry but keeps it below the same parent (i.e., renames the entry), only the `write` permission is required. This is true regardless of whether the entry being renamed is a leaf entry or has subordinate entries.
import	See the description for the `export` permission.
proxy	Indicates that the access control rule should apply to operations that attempt to use an alternate authorization identity (for example, operations that include a proxied authorization request control, an intermediate client request control with an alternate authorization identity, or a client that has authenticated with a SASL mechanism that allows an alternate authorization identify to be specified).
read	Indicates that the access control rule should apply to search result entries returned by the server.
search	Indicates that the access control rule should apply to search operations with a non-base scope.
selfwrite	Indicates that the access control rule should apply to operations in which a user attempts to add or remove his or her own DN to the values for an attribute (for example, whether users may add or remove themselves from groups).
write	Indicates that the access control rule should apply to modify and modify DN operations.
all	An aggregate permission that includes all other permissions except “proxy.” This is equivalent to providing a permission of “add, compare, delete, read, search, selfwrite, write, export, and import.”

Bind rules

The Bind Rules indicate whether an access control rule should apply to a given requester. The syntax for the target keyword is shown below. The *keyword* specifies the type of target element. The expression

specifies the items that is targeted by the access control rule. The operator is either equals ("=") or not-equals ("!="). The semi-colon delimiter symbol (";") is required after the end of the final bind rule.

```
keyword [=||!= ] expression;
```

Multiple bind rules can be combined using Boolean operations (AND, OR, NOT) for more access control precision. The standard Boolean rules for evaluation apply: innermost to outer parentheses first, left to right expressions, NOT before AND or OR. For example, an ACI that includes the following bind rule targets all users who are not `uid=admin,dc=example,dc=com` and use simple authentication.

```
(userdn!="ldap:///uid=admin,dc=example,dc=com" and authmethod="simple");
```

The following bind rule targets the `uid=admin,dc=example,dc=com` and authenticates using SASL EXTERNAL or accesses the server from a loopback interface.

```
(userdn="ldap:///uid=admin,dc=example,dc=com and (authmethod="SSL" or
 ip="127.0.0.1"));
```

The following keywords are supported for use in the bind rule portion of ACIs:

Summary of Bind Rule Keywords

Bind Rule Keyword	Description

authmethod	Indicates that the requester's authentication method should be taken into account when determining whether the access control rule should apply to an operation. Wildcards are not allowed in this expression. The keyword's syntax is as follows:
	
	authmethod = *method*
	
	where *method* is one of the following representations:
	▪ none
	▪ simple. Indicates that the client is authenticated to the server using a bind DN and password.
	▪ ssl. Indicates that the client is authenticated with an SSL/TLS certificate (for example, via SASL EXTERNAL), and not just over a secure connection to the server.
	▪ sasl {sasl_mechanism}. Indicates that the client is authenticated to the server using a specified SASL Mechanism.
	
	The following example allows users who authenticate with an SSL/TLS certificate (for example., via SASL EXTERNAL) to update their own entries:
	
	`aci: (targetattr="*")`
	`(version 3.0; acl "Allow users to update their own entries";`
	`allow (write) (userdn="ldap:///self" and authmethod="ssl");)`

Bind Rule Keyword	Description

dayofweek	Indicates that the day of the week should be taken into account when determining whether the access control rule should apply to an operation. Wildcards are not allowed in this expression. Multiple day of week values may be separated by commas. The keyword's syntax is as follows:
	
	dayofweek = *day1, day2, ...*
	
	where *day* is one of the following representations:
	
	▪ sun
	▪ mon
	▪ tues
	▪ wed
	▪ thu
	▪ fri
	▪ sat
	
	The following example allows users who authenticate with an SSL/TLS certificate (for example., via SASL EXTERNAL) on weekdays to update their own entries:
	
	`aci: (targetattr="*")`
	`  (version 3.0; acl "Allow users to update their own entries";`
	`   allow (write) (dayofweek!="sun,sat" and userdn="ldap:///`
	`self"`
	`   and authmethod="ssl");)`

dns	Indicates that the requester's DNS-resolvable host name should be taken into account when determining whether the access control rule should apply to an operation. Wildcards are allowed in this expression. Multiple DNS patterns may be separated by commas. The keyword's syntax is as follows:
	
	dns = *dns-host-name*
	
	The following example allows users on host name `server.example.com` to update their own entries:
	
	`aci: (targetattr="*")`
	`  (version 3.0; acl "Allow users to update their own entries";`
	`   allow (write) (dns="server.example.com" and userdn="ldap:///`
	`self");)`

Bind Rule Keyword	Description

groupdn	Indicates that the requester's group membership should be taken into account when determining whether the access control rule should apply to any operation. Wildcards are not allowed in this expression.
	
	groupdn [= || !=] "ldap:///groupdn [|| ldap:///groupdn] ..."
	
	The following example allows users in the managers group to update their own entries:
	
	`aci: (targetattr="*")`
	`  (version 3.0; acl "Allow users to update their own entries";`
	`   allow (write)`
	`   (groupdn="ldap:///`
	`cn=managers,ou=groups,dc=example,dc=com");)`

ip	Indicates that the requester's IP address should be taken into account when determining whether the access control rule should apply to an operation. Wildcards are allowed in this expression. Multiple IP address patterns may be separated by commas. The keyword's syntax is as follows:
	
	ip `[ = || != ]` *`ipAddressList`*
	
	where *ipAddressList* is one of the following representations:
	▪ A specific IPv4 address: 127.0.0.1
	▪ An IPv4 address with wildcards to specify a subnetwork: 127.0.0.*
	▪ An IPv4 address or subnetwork with subnetwork mask: 123.4.5.0+255.255.255.0
	▪ An IPv4 address range using CIDR notation: 123.4.5.0/24
	▪ An IPv6 address as defined by RFC 2373.
	
	The following example allows users on 10.130.10.2 and localhost to update their own entries:
	
	`aci: (targetattr="*")`
	`  (version 3.0; acl "Allow users to update their own entries";`
	`   allow (write) (ip="10.130.10.2,127.0.0.1" and`
	` userdn="ldap:///self");)`

Bind Rule Keyword	Description

oauthscope	Indicates that the scopes associated with any OAuth 2.0 access token presented by a SCIMv2 client should be taken into account when determining whether the access control rule should apply to an operation. The keyword's syntax is as follows:
	
	`oauthscope [ = || != ] "scopeIdentifier"`
	
	where *scopeIdentifier* is one of the following:
	▪ The name of a single scope to match. The name will be treated as case-sensitive.
	▪ A substring assertion that contains one or more asterisks as wildcards.
	▪ A single asterisk by itself, which will match any scope.
	
	The following example will grant all rights to any client that presented an OAuth 2.0 token that is associated with the "scim_admin" scope:
	
	`aci: (targetattr="*")`
	`  (version 3.0; acl "Full rights for users with the scim_admin OAuth 2.0 scope";`
	`  allow (all) oauthscope="scim_admin";)`

timeofday	Indicates that the time of day should be taken into account when determining whether the access control rule should apply to an operation. Wildcards are not allowed in this expression. The keyword's syntax is as follows:
	
	`timeofday [ = || != || >= || > || <= || < ] time`
	
	where *time* is one of the following representations:
	▪ 4-digit 24-hour time format (0000 to 2359, where the first two digits represent the hour of the day and the last two represent the minute of the hour)
	▪ Wildcards are not allowed in this expression
	
	The following example allows users to update their own entries if the request is received before 12 noon.
	
	`aci: (targetattr="*")`
	`  (version 3.0; acl "Allow users who authenticate before noon to update their own entries";`
	`  allow (write) (timeofday<1200 and userdn="ldap:///self" and authmethod="simple");)`

Bind Rule Keyword	Description

oauthscope	Indicates that the scopes associated with any OAuth 2.0 access token presented by a SCIMv2 client should be taken into account when determining whether the access control rule should apply to an operation. The keyword's syntax is as follows:
	
	`oauthscope [ = || != ] "scopeIdentifier"`
	
	where *scopeIdentifier* is one of the following:
	
	▪ The name of a single scope to match. The name will be treated as case-sensitive.
	▪ A substring assertion that contains one or more asterisks as wildcards.
	▪ A single asterisk by itself, which will match any scope.
	
	The following example will grant all rights to any client that presented an OAuth 2.0 token that is associated with the "scim_admin" scope:
	
	`aci: (targetattr="*")`
	`  (version 3.0; acl "Full rights for users with the scim_admin OAuth 2.0 scope";`
	`    allow (all) oauthscope="scim_admin";)`

Bind Rule Keyword	Description

userattr	Indicates that the requester's relation to the value of the specified attribute should be taken into account when determining whether the access control rule should apply to an operation. A bindType value of `USERDN` indicates that the target attribute should have a value which matches the DN of the authenticated user. A bindType value of `GROUPDN` indicates that the target attribute should have a value which matches the DN of a group in which the authenticated user is a member. A bindType value of `LDAPURL` indicates that the target attribute should have a value that is an LDAP URL whose criteria matches the entry for the authenticated user. Any value other than `USERDN`, `GROUPDN`, or `LDAPURL` is expected to be present in the target attribute of the authenticated user's entry. The keyword's syntax is as follows:
	
	`userattr = attrName# [ bindType || attrValue ]`
	
	where:
	▪ *attrName* = name of the attribute for matching
	▪ *bindType* = USERDN, GROUPDN, LDAPURL
	▪ *attrValue* = an attribute value. Note that the *attrVALUE* of the attribute must match on both the bind entry and the target of the ACI.
	
	The following example allows a manager to change employee's entries. If the bind DN is specified in the *manager* attribute of the targeted entry, the bind rule is evaluated to TRUE.
	
	`aci: (targetattr="*")`
	`  (version 3.0; acl "Allow a manager to change employee entries";`
	`   allow (write) userattr="manager#USERDN";)`
	
	The following example allows any member of a group to change employee's entries. If the bind DN is a member of the group specified in the *allowEditors* attribute of the targeted entry, the bind rule is evaluated to TRUE.
	
	`aci: (targetattr="*")`
	`  (version 3.0; acl "Allow allowEditors to change employee entries";`
	`   allow (write) userattr="allowEditors#GROUPDN";)`
	
	The following example allows a user's manager to edit that user's entry and any entries below the user's entry up to two levels deep. You can specify up to five levels (0, 1, 2, 3, 4) below the targeted entry, with zero (0) indicating the targeted entry.
	
	`aci: (targetattr="*")`
	`  (version 3.0; acl "Allow managers to change employees entries two levels below";`
	`   allow (write) userattr="parent[0,1,2].manager#USERDN";)`
	
	The following example allows any member of the engineering department to update any other member of the engineering department at or below the specified ACI.
	
	`aci: (targetattr="*")`
	`  (version 3.0; acl "Allow any member of Eng Dept to update any other member of the`
	`   enginering department at or below the ACI";`
	`   allow (write) userattr="department#ENGINEERING";)`
	
	The following example allows an entry to be updated by any user whose entry matches the criteria defined in the LDAP URL contained in the `allowedEditorCriteria` attribute of the target entry.
	
	`aci: (targetattr="*")`
	`  (version 3.0; acl "Allow a user that matches the filter to change entries";`

Bind Rule Keyword	Description

userdn	Indicates that the user's DN should be taken into account when determining whether the access control rule should apply to an operation. The keyword's syntax is as follows:
	
	`userdn [ = || != ] "ldap:///value [ || "ldap:///value ..."]`
	
	where *value* is one of the following representations:
	
	▪ The DN of the target user
	▪ A value of `anyone` to match any client, including unauthenticated clients.
	▪ A value of `all` to match any authenticated client.
	▪ A value of `parent` to match the client authenticated as the user defined in the immediate parent of the target entry.
	▪ A value of `self` to match the client authenticated as the user defined in the target entry.
	
	If the value provided is a DN, then that DN may include wildcard characters to define patterns. A single asterisk will match any content within the associated DN component, and two consecutive asterisks may be used to match zero or more DN components.
	
	The following example allows users to update their own entries:
	
	`aci: (targetattr="*")`
	`  (version 3.0; acl "Allow users to update their own entries";`
	`  allow (write) userdn="ldap:///self";)`

Access token validators

Access token validators verify the tokens that HTTP client applications submit when they request access to protected resources and associate each token with an identity stored in the directory server.

To authenticate to PingDirectory Server's HTTP services, clients use OAuth 2 bearer token authentication to present an access token in the HTTP Authorization request header. To process the incoming access tokens, PingDirectory Server uses access token validators, which determine whether to accept an access token and translate it into a set of properties, called *claims*, which PingDirectory Server's HTTP services use to make access control decisions.

Most access tokens identify a user as its subject using the sub claim. Access token validators can retrieve the token subject's attributes from the directory using an identity mapper, which correlates the access token subject to an LDAP entry.

Access token validators are used by the following services:

- Directory REST API
- SCIM 2
- Delegated Admin
- Consent API

In addition, PingDirectory Server can be configured to accept access tokens provided by LDAP clients using the OAUTHBEARER SASL authentication method.

About access token validator processing

You can configure any number of access token validators for PingDirectoryProxy Server.

Each access token validator possesses an evaluation order index, an integer that determines its processing priority when multiple access token validators are configured. Lower values are processed before higher values.

1. If an incoming HTTP request contains an access token, the token is sent to the access token validator with the lowest evaluation order index.
2. The access token validator validates the access token. Validation logic varies by access token validator type, but the validator generally verifies the following information:
   - A trusted source issued the token.
   - The token is not expired.
3. If the access token contains a subject, the access token validator uses its identity mapper to find a matching LDAP entry.
4. If the access token validator is unable to validate the access token, it passes the token to the access token validator with the next lowest evaluation order index, and the previous two steps are repeated.
5. HTTP request processing continues, and the policy request is sent to the HTTP service, such as the Directory REST API, for further evaluation.
6. Using either the access token claims parsed by the access token validator or the LDAP entry found by the identity mapper, the HTTP service determines whether the request should be accepted and which access control rules should be applied. This access control behavior varies by each HTTP service.

> ⓘ **Note:**
>
> Access tokens issued using the OAuth 2 client credentials grant type are issued directly to a client and do not contain a subject. Such tokens cannot be accepted by PingDirectory Server.

Access token validator types

PingDirectory Server works with a variety of access token validators.

PingFederate access token validator
To verify the access tokens that a PingFederate authorization server issues, the PingFederate access token validator uses HTTP to submit the tokens to PingFederate Server's token introspection endpoint.

This step allows the authorization server to determine whether a token is valid.

> ⓘ **Note:**
>
> Access tokens issued using the OAuth 2 client credentials grant type are issued directly to a client and do not contain a subject. Such tokens cannot be accepted by PingDirectory Server.

Because this step requires an outgoing HTTP request to the authorization server, the PingFederate access token validator might perform slower than other access token validator types. The validation result is guaranteed to be current, which is an important consideration if the authorization server permits the revocation of access tokens.

Before attempting to use a PingFederate access token validator, create a client that represents the access token validator in the PingFederate configuration. This client must use the Access Token Validation grant type.

Example configuration

In PingFederate, create a client with the following properties:

- Client ID: Ping Identity
- Client authentication: Client Secret
- Allowed grant types: Access Token Validation

Take note of the client secret that is generated for the client, and use the PingDirectory Server's **`dsconfig`** command to create an access token validator, as shown.

```
# Create an identity mapper that expects the token subject to be a uid
dsconfig create-identity-mapper \
 --mapper-name "User ID Identity Mapper" \
 --type exact-match \
 --set enabled:true \
 --set match-attribute:uid \
 --set match-base-dn:ou=people,dc=example,dc=com
# Change the host name and port below, as needed
dsconfig create-external-server \
  --server-name "PingFederate External Server" \
  --type http \
  --set base-url:https://example.com:9031
# Create the Access Token Validator
dsconfig create-access-token-validator \
  --validator-name "PingFederate Access Token Validator" \
  --type ping-federate \
  --set enabled:true \
  --set "authorization-server:PingFederate External Server" \
  --set client-id:PingDataGovernance \
  --set "client-secret:<client secret>"
  --set evaluation-order-index:2000
  --set "identity-mapper:User ID Identity Mapper"
```

Replace `<client secret>` with the client secret value generated by the PingFederate client.

JWT access token validator

The JWT access token validator verifies access tokens that are encoded in JSON Web Token (JWT) format, which can be signed in JSON web signature (JWS) format or signed and encrypted in JSON web encryption (JWE) format.

The JWT access token validator inspects the JWT token without presenting it to an authorization server for validation. Because the JWT access token validator does not make a token introspection request for every access token that it processes, it performs faster than the PingFederate access token validator. The access token is self-validated however, so the JWT access token validator cannot determine whether the token has been revoked.

Supported JWS/JWE features

For signed tokens, the JWT access token validator supports the following JWT web algorithm (JWA) types:

- RS256
- RS384
- RS512
- ES256
- ES384
- ES512

For encrypted tokens, the JWT access token validator supports the following key-encryption algorithms:

- RSA-OAEP
- ECDH-ES
- ECDH-ES+A128KW
- ECDH-ES+A192KW
- ECDH-ES+A256KW

For encrypted tokens, the JWT access token validator supports the following content-encryption algorithms:

- A128CBC-HS256
- A192CBC-HS384
- A256CBC-HS512

The JWT access token validator configuration defines three *allow lists* for the JWS/JWE signing and encryption algorithms that it will accept. You should customize these allow lists to reflect only the signing and encryption algorithms used by your access token issuer and no others. Doing so minimizes the access token validator's security threat surface.

Configure these allow lists using the following configuration properties:

- `allowed-signing-algorithm`

  Specifies the signing algorithms that the access token validator accepts.
- `allowed-key-encryption-algorithm`

  Specifies the key-encryption algorithms that the access token validator accepts.
- `allowed-content-encryption-algorithm`

  Specifies the content-encryption algorithms that the access token validator accepts.

Handling signed tokens
All access tokens the JWT access token validator handles must be cryptographically signed by the token issuer. The JWT access token validator validates a token's signature using a public signing key provided by the issuer.

Steps

- Configure the JWT access token validator with the issuer's public signing key in one of two ways:
  - Store the public key as a trusted certificate in PingDirectory Server's local configuration using the `trusted-certificate` property.
  - Provide the issuer's JWKS (JSON Web Key Set) endpoint using the `jwks-endpoint-path` property. The JWT access token validator then retrieves the issuer's public keys when it initializes. This method ensures that the JWT access token validator uses updated copies of the issuer's public keys.

Example: Use a locally configured trusted certificate
The following example configures a JWT access token validator to use a locally stored public signing certificate to validate access token signatures. The signing certificate is assumed to have been obtained out of band and must be a PEM-encoded X.509v3 certificate.

```
# Create an identity mapper that expects the token subject to be a uid
dsconfig create-identity-mapper \
 --mapper-name "User ID Identity Mapper" \
 --type exact-match \
 --set enabled:true \
 --set match-attribute:uid \
 --set match-base-dn:ou=people,dc=example,dc=com

# Add the public signing certificate to the server configuration
dsconfig create-trusted-certificate \
 --certificate-name "JWT Signing Certificate" \
 --set "certificate</path/to/signing-certificate.pem"

# Create the Access Token Validator
dsconfig create-access-token-validator \
 --validator-name "JWT Access Token Validator" \
 --type jwt \
 --set enabled:true \
 --set evaluation-order-index:1000 \
 --set allowed-signing-algorithm:RS256 \
 --set "trusted-certificate:JWT Signing Certificate"
 --set "identity-mapper:User ID Identity Mapper"
```

Example: Use the issuer's JWKS endpoint

The following example configures a JWT access token validator to retrieve public keys from a PingFederate authorization server's JWKS endpoint.

```
# Create an identity mapper that expects the token subject to be a uid
dsconfig create-identity-mapper \
 --mapper-name "User ID Identity Mapper" \
 --type exact-match \
 --set enabled:true \
 --set match-attribute:uid \
 --set match-base-dn:ou=people,dc=example,dc=com

# Change the host name and port below, as needed
dsconfig create-external-server \
 --server-name "PingFederate External Server" \
 --type http \
 --set base-url:https://example.com:9031

# Create the Access Token Validator
dsconfig create-access-token-validator \
 --validator-name "JWT Access Token Validator" \
 --type jwt \
 --set enabled:true \
 --set evaluation-order-index:1000 \
 --set allowed-signing-algorithm:RS256 \
 --set "authorization-server:PingFederate External Server" \
 --set jwks-endpoint-path:/ext/oauth/jwks
 --set "identity-mapper:User ID Identity Mapper"
```

Handling encrypted tokens

Optionally, you can configure the JWT access token validator to accept encrypted access tokens. To do this, you must configure the access token validator with a private/public key pair and provide the public key to the token issuer.

Steps

1. Create an encryption key pair.
2. Create the JWT access token validator.
3. Export the public encryption key from PingDataGovernance Server and provide it to your token issuer.

   Without this public encryption key, the issuer cannot encrypt tokens that can be decrypted by the JWT access token validator.

   You can run **`dsconfig`** to copy the public key to a file, or you can copy the value of the key pair's `certificate-chain` property in the Administrative Console.

Example

The following example configures a JWT access token validator to handle access tokens signed and encrypted using elliptic curve algorithms. For RSA signing and encryption algorithms, the configuration is very similar, but you would choose different values for the `allowed-signing-algorithm` and `allowed-encryption-algorithm` properties.

1. Create an encryption key pair.

```
# Create an encryption key pair
 dsconfig create-key-pair \
 --pair-name "JWT Elliptic Curve Encryption Key Pair" \
```

```
 --set key-algorithm:EC_256
```

2. Create the JWT access token validator.

```
# Create an identity mapper that expects the token subject to be a uid
dsconfig create-identity-mapper \
 --mapper-name "User ID Identity Mapper" \
 --type exact-match \
 --set enabled:true \
 --set match-attribute:uid \
 --set match-base-dn:ou=people,dc=example,dc=com

# Change the host name and port below, as needed
dsconfig create-external-server \
 --server-name "PingFederate External Server" \
 --type http \
 --set base-url:https://example.com:9031

# Create the Access Token Validator
dsconfig create-access-token-validator \
 --validator-name "JWT Access Token Validator" \
 --type jwt \
 --set enabled:true \
 --set evaluation-order-index:1000 \
 --set allowed-signing-algorithm:ES256 \
 --set "authorization-server:PingFederate External Server" \
 --set jwks-endpoint-path:/ext/oauth/jwks \
 --set "encryption-key-pair:JWT Elliptic Curve Encryption Key Pair" \
 --set allowed-key-encryption-algorithm:ECDH_ES
 --set "identity-mapper:User ID Identity Mapper"
```

3. Export the public encryption key from PingDataGovernance Server and provide it to your token issuer.

   The following command copies the key to a file.

```
 dsconfig get-key-pair-prop \
 --pair-name "JWT Elliptic Curve Encryption Key Pair" \
 --property certificate-chain \
 --no-prompt \
 --script-friendly > jwt-public-encryption-key.pem
```

Mock access token validator

A mock access token validator is a special access token validator type used for development or testing purposes.

A mock access token validator accepts arbitrary tokens without validating whether a trusted source issued them. This approach allows a developer or tester to make bearer token-authenticated requests without first setting up an authorization server.

Mock access tokens are formatted as plain-text JSON objects using standard JSON web token (JWT) claims.

Always provide the boolean `active` claim when creating a mock token. If this value is `true`, the token is accepted. If this value is `false`, the token is rejected.

If the `sub` claim is provided, a token owner lookup populates the `TokenOwner` policy request attribute, as with the other access token validator types.

The following example cURL command provides a mock access token in an HTTP request.

```
curl -k -X GET https://localhost:8443/scim/v2/Me -H 'Authorization:
 Bearer {"active": true, "sub":"user.3", "scope":"email profile",
 "client":"client1"}'
```

> ⓘ **Important:**
>
> Never use mock access token validators in a production environment because they do not verify whether a trusted source issued an access token.

Example configuration

The configuration for a mock access token validator resembles the configuration for a JWT access token validator. However, the JSON web signature (JWS) signatures require no configuration because mock tokens are not authenticated.

```
# Create an identity mapper that expects the token subject to be a uid
 dsconfig create-identity-mapper \
 --type exact-match \
 --set enabled:true \
 --set match-attribute:uid \
 --set match-base-dn:ou=people,dc=example,dc=com

# Create the Access Token Validator
dsconfig create-access-token-validator \
  --validator-name "Mock Access Token Validator" \
  --type mock --set enabled:true \
  --set evaluation-order-index:9999
  --set "identity-mapper:User ID Identity Mapper"
```

Third-party access token validator
Third-party access token validator

To create custom access token validators, use the Server SDK.

Working with targets

The following section presents a detailed look and examples of the target ACI keywords: `target`, `targetattr`, `targetfilter`, `targattrfilters`, `targetscope`, `targetcontrol`, and `extop`.

target

The `target` keyword indicates that the ACI should apply to one or more entries at or below the specified distinguished name (DN). The target DN must be equal or subordinate to the DN of the entry in which the ACI is placed. For example, if you place the ACI at the root of `ou=People,dc=example,dc=com`, you can target the DN, `uid=user.1,ou=People,dc=example,dc=com` within your ACI rule. The DN must meet the string representation specification of distinguished names, outlined in RFC 4514, and requires that special characters be properly escaped.

The `target` clause has the following format, where DN is the distinguished name of the entry or branch:

```
(target = ldap:///
                  DN
               )
```

For example, to target a specific entry, you would use a clause such as the following:

```
(target = ldap:///uid=john.doe,ou=People,dc=example,dc=com)
```

Note that, in general, specifying a target DN is not recommended. It is better to have the ACI defined in that entry and omit the `target` element altogether. For example, although you can have `(target="ldap:///uid=john.doe,ou=People,dc=example,dc=com)` in any of the `dc=example,dc=com` or `ou=People` entries, it is better for it to be defined in the `uid=john.doe` entry and not explicitly include the `target` element.

The expression allows for the "not equal" (!=) operator to indicate that all entries within the scope of the given branch that do NOT match the expression be targeted for the ACI. Thus, the following expression targets all entries within the subtree that do not match `uid=john.doe`.

```
(target != ldap:///uid=john.doe,ou=People,dc=example,dc=com)
```

The `target` keyword also supports the use of asterisk (*) characters as wildcards to match elements within the distinguished name. The following target expression matches all entries that contains and begins with "john.d, " so that entries like "`john.doe,ou=People,dc=example,dc=com`," and "`john.davies,ou=People,dc=example,dc=com`" would match.

```
(target = ldap:///uid=john.d*,ou=People,dc=example,dc=com)
```

The following target expression matches all entries whose DN begins with "john.d," and matches the `ou` attribute. Entries like "`john.doe,ou=People,dc=example,dc=com`," and "`john.davies,ou=asia-branch,dc=example,dc=com`" would match.

```
(target = ldap:///uid=john.d*,ou=*,dc=example,dc=com)
```

Another example of a complete ACI targets the entries in the `ou=People,dc=example,dc=com` branch and the entries below it, and grants the users the privilege to modify all of their user attributes within their own entries.

```
aci:(target="ldap:///ou=People,dc=example,dc=com")
  (targetattr="*")
  (version 3.0; acl "Allow all the ou=People branch to modify their own
 entries";
  allow (write) userdn="ldap:///self";)
```

targetattr

The `targetattr` keyword targets the attributes for which the access control instruction should apply. There are four general forms that it can take in the PingDirectoryProxy Server:

- **(targetattr="*")**. Indicates that the access control rule applies to all user attributes. Operational attributes will not automatically be included in this set.
- **(targetattr="+")**. Indicates that the access control rule applies to all operational attributes. User attributes will not automatically be included in this set.
- **(targetattr="attr1||attr2||attr3||...||attrN")**. Indicates that the access control rule applies only to the named set of attributes.
- **(targetattr!="attr1||attr2||attr3||...||attrN")**. Indicates that the access control rule applies to all user attributes except the named set of attributes. It will not apply to any operational attributes.

The targeted attributes can be classified as user attributes and operational attributes. User attributes define the actual data for that entry, while operational attributes provide additional metadata about the entry that can be used for informational purposes, such as when the entry was created, last modified and by whom. Metadata can also include attributes specifying which password policy applies to the user, or overrided default constraints like size limit, time limit, or look-through limit for that user.

The PingDirectoryProxy Server distinguishes between these two types of attributes in its access control implementation. The Directory Proxy Server does not automatically grant any access at all to operational attributes. For example, the following clause applies only to user attributes and not to operational attributes:

```
(targetattr="*")
```

You can also target multiple attributes in the entry. The following clause targets the common name (cn), surname (sn) and state (st) attribute:

```
(targetattr="cn||sn||st")
```

You can use the "+" symbol to indicate that the rule should apply to all operational attributes, as follows:

```
(targetattr="+")
```

To include all user and all operational attributes, you use both symbols, as follows:

```
(targetattr="*||+")
```

If there is a need to target a specific operational attribute rather than all operational attributes, then it can be specifically included in the values of the `targetattr` clause, as follows:

```
(targetattr="ds-rlim-size-limit")
```

Or if you want to target all user attributes and a specific operational attribute, then you can use them in the `targetattr` clause, as follows:

```
(targetattr="*||ds-rlim-size-limit")
```

The following ACIs are placed on the `dc=example,dc=com` tree and allows any user anonymous read access to all entries except the `userPassword` attribute. The second ACI allows users to update their own contact information. The third ACI allows the `uid=admin` user full access privileges to all user attributes in the `dc=example,dc=com` subtree.

```
aci: (targetattr!="userPassword")(version 3.0; acl "Allow anonymous
   read access for anyone"; allow (read,search,compare) userdn="ldap:///
anyone";)
aci: (targetattr="telephonenumber||street||homePhone||l||st")
  (version 3.0; acl "Allow users to update their own contact info";
   allow (write) userdn="ldap:///self";)
aci: (targetattr="*")(version 3.0; acl "Grant full access for the admin
 user";
  allow (all) userdn="ldap:///uid=admin,dc=example,dc=com";)
```

An important note must be made when assigning access to user and operational attributes, which can be outlined in an example to show the implications of the Directory Proxy Server not distinguishing between these attributes. It can be easy to inadvertently create an access control instruction that grants far more capabilities to a user than originally intended. Consider the following example:

```
aci: (targetattr!="uid||employeeNumber")
  (version 3.0; acl "Allow users to update their own entries";
    allow (write) userdn="ldap:///self";)
```

This instruction is intended to allow a user to update any attribute in his or her own entry with the exception of `uid` and `employeeNumber`. This ACI is a very common type of rule and seems relatively harmless on the surface, but it has very serious consequences for a Directory Proxy Server that does not distinguish between user attributes and operational attributes. It allows users to update operational attributes in their own entries, and could be used for a number of malicious purposes, including:

- A user could alter password policy state attributes to become exempt from password policy restrictions.
- A user could alter resource limit attributes and bypass size limit, time limit, and look-through-limit constraints.
- A user could add access control rules to his or her own entry, which could allow them to make their entry completely invisible to all other users including administrators granted full rights by access control rules, but excluding users with the `bypass-acl` privilege, allow them to edit any other attributes in their

own entry including those excluded by rules like `uid` and `employeeNumber` in the example above, or add, modify, or delete any entries below his or her own entry.

Because the PingDirectoryProxy Server does not automatically include operational attributes in the target attribute list, these kinds of ACIs do not present a security risk for it. Also note that users cannot add ACIs to any entries unless they have the `modify-acl` privilege.

Another danger in using the `(targetattr!="x")` pattern is that two ACIs within the same scope could have two different `targetattr` policies that cancel each other out. For example, if one ACI has `(targetattr!="cn||sn")` and a second ACI has `(targetattr!="userPassword")`, then the net effect is `(targetattr="*")`, because the first ACI inherently allows `userPassword`, and the second allows `cn` and `sn`.

targetfilter

The `targetfilter` keyword targets all attributes that match results returned from a filter. The `targetfilter` clause has the following syntax:

```
(targetfilter =
                    ldap_filter
                )
```

For example, the following clause targets all entries that contain "ou=engineering" attribue:

```
(targetfilter = "(ou=engineering)")
```

You can only specify a single filter, but that filter can contain multiple elements combined with the OR operator. The following clause targets all entries that contain "ou=engineering," "ou=accounting," and "ou=marketing."

```
(targetfilter = "(|(ou=engineering)(ou=accounting)(ou=marketing)")
```

The following example allows the user, `uid=eng-mgr`, to modify the `departmentNumber`, `cn`, and `sn` attributes for all entries that match the filter `ou=engineering`.

```
aci:(targetfilter="(ou=engineering)")
  (targetattr="departmentNumber||cn||sn")
  (version 3.0; acl "example"; allow (write)
   userdn="ldap:///uid=eng-mgr,dc=example,dc=com";)
```

targattrfilters

The `targattrfilters` keyword targets specific attribute *values* that match a filtered search criteria. This keyword allows you to set up an ACI that grants or denies permissions on an attribute value if that value meets the filter criteria. The `targattrfilters` keyword applies to individual values of an attribute, not to the whole attribute. The keyword also allows the use of wildcards in the filters.

The keyword clause has the following formats:

```
(target = "add=attr1:Filter1 && attr2:Filter2... && attrn:FilterN,
del=attr1:Filter1 && attr2:Filter2 ... && attrN:FilterN" )
```

where

- **add** represents the operation of adding an attribute value to the entry
- **del** represents the operation of removing an attribute value from the entry
- **attr1, attr2... attrN** represents the targeted attributes
- **filter1, filter2 ... filterN** represents filters that identify matching attribute values

The following conditions determine when the attribute must satisfy the filter:

- When adding or deleting an entry containing an attribute targeted a `targattrfilters` element, each value of that attribute must satisfy the corresponding filter.

- When modifying an entry, if the operation adds one or more values for an attribute targeted by a `targattrfilters` element, each value must satisfy the corresponding filter. If the operation deletes one or more values for a targeted attribute, each value must satisfy the corresponding filter.
- When replacing the set of values for an attribute targeted by a `targattrfilters` element, each value removed must satisfy the delete filters, and each value added must satisfy the add filters.

The following example allows any user who is part of the `cn=directory server admins` group to add the `soft-delete-read` privilege.

```
aci:(targattrfilter="add=ds-privilege-name:(ds-privilege-name=soft-delete-
read)")
  (version 3.0; acl "Allow members of the directory server admins group to
 grant the
   soft-delete-read privilege"; allow (write)
   groupdn="ldap:///cn=directory server admins,ou=group,dc=example,dc=com";)
```

targetscope

The `targetscope` keyword is used to restrict the scope of an access control rule. By default, ACIs use a subtree scope, which means that they are applied to the target entry (either as defined by the target clause of the ACI, or the entry in which the ACI is define if it does not include a target), and all entries below it. However, adding the `targetscope` element into an access control rule can restrict the set of entries to which it applies.

The following `targetscope` keyword values are allowed:

- **base**. Indicates that the access control rule should apply only to the target entry and not to any of its subordinates.
- **onelevel**. Indicates that the access control rule should apply only to entries that are the immediate children of the target entry and not to the target entry itself, nor to any subordinates of the immediate children of the target entry.
- **subtree**. Indicates that the access control rule should apply to the target entry and all of its subordinates. This is the default behavior if no `targetscope` is specified.
- **subordinate**. Indicates that the access control rule should apply to all entries below the target entry but not the target entry itself.

The following ACI targets all users to view the operational attributes (`supportedControl`, `supportedExtension`, `supportedFeatures`, `supportedSASLMechanisms`, `vendorName`, and `vendorVersion`) present in the root DSE entry. The `targetscope` is base to limit users to view only those attributes in the root DSE.

```
aci: (target="ldap:///")(targetscope="base")
    (targetattr="supportedControl||supportedExtension||
     supportedFeatures||supportedSASLMechanisms||vendorName||vendorVersion")
    (version 3.0; acl "Allow users to view Root DSE Operational Attributes";
     allow (read,search,compare) userdn="ldap:///anyone")
```

targetcontrol

The `targetcontrol` keyword is used to indicate whether a given request control can be used by those users targeted in the ACI. Multiple OIDs can be provided by separating them with the two pipe characters (optionally surrounded by spaces). Wildcards are not allowed when specifying control OIDs.

The following ACI example shows the controls required to allow an administrator to use and manage the Soft-Delete feature. The Soft Delete Request Control allows the user to soft-delete an entry, so that it could be undeleted at a later time. The Hard Delete Request Control allows the user to permanently remove an entry or soft-deleted entry. The Undelete Request Control allows the user to undelete a currently soft-deleted entry. The Soft-Deleted Entry Access Request Control allows the user to search for any soft-deleted entries in the server.

```
aci: (targetcontrol="1.3.6.1.4.1.30221.2.5.20||1.3.6.1.4.1.30221.2.5.22||
```

```
   1.3.6.1.4.1.30221.2.5.23||1.3.6.1.4.1.30221.2.5.24")
  (version 3.0; acl "Allow admins to use the Soft Delete Request Control,
   Hard Delete Request Control,Undelete Request Control, and
   Soft-deleted entry access request control";
   allow (read) userdn="ldap:///uid=admin,dc=example,dc=com";)
```

extOp

The `extop` keyword can be used to indicate whether a given extended request operation can be used. Multiple OIDs can be provided by separating them with the two pipe characters (optionally surrounded by spaces). Wildcards are not allowed when specifying extended request OIDs.

The following ACI allows the `uid=user-mgr` to use the Password Modify Request (i.e., OID=1.3.6.1.4.1.4203.1.11.1) and the StartTLS (i.e., OID=1.3.6.1.4.1.1466.20037) extended request OIDs.

```
aci:(extop="1.3.6.1.4.1.4203.1.11.1 || 1.3.6.1.4.1.1466.20037")
  (version 3.0; acl "Allows the mgr to use the Password Modify Request and
 StartTLS;
   allow(read) userdn="ldap:///uid=user-mgr,ou=people,dc=example,dc=com";)
```

Examples of common access control rules

This section provides a set of examples that demonstrate access controls that are commonly used in your environment. Note that to be able to alter access control definitions in the server, a user must have the `modify-acl` privilege as discussed later in this chapter.

Administrator access

The following ACI can be used to grant any member of the "`cn=admins,ou=groups,dc=example,dc=com`" group to add, modify and delete entries, reset passwords and read operational attributes such as `isMemberOf` and password policy state:

```
aci: (targetattr="+")(version 3.0; acl "Administrators can read, search or
 compare operational attributes";
allow (read,search,compare) groupdn="ldap:///
cn=admins,ou=groups,dc=example,dc=com";)
aci: (targetattr="*")(version 3.0; acl "Administrators can add, modify and
 delete entries";
allow (all) groupdn="ldap:///cn=admins,ou=groups,dc=example,dc=com";)
```

Anonymous and authenticated access

The following ACI allow anonymous read, search and compare on select attributes of `inetOrgPerson` entries while authenticated users can access several more. The authenticated user will inherit the privileges of the anonymous ACI. In addition, the authenticated user can change `userPassword`:

```
aci: (targetattr="objectclass || uid || cn || mail || sn || givenName")
(targetfilter="(objectClass=inetorgperson)")
(version 3.0; acl "Anyone can access names and email addresses of entries
 representing people";
allow (read,search,compare) userdn="ldap:///anyone";)
aci: (targetattr="departmentNumber || manager || isMemberOf")
(targetfilter="(objectClass=inetorgperson)")
(version 3.0; acl "Authenticated users can access these fields for entries
 representing people";
allow (read,search,compare) userdn="ldap:///all";)
aci: (targetattr="userPassword")(version 3.0; acl "Authenticated users can
 change password";
allow (write) userdn="ldap:///all";)
```

If no unauthenticated access should be allowed to the Directory Server, the preferred method for preventing unauthenticated, or anonymous access is to set the Global Configuration property `reject-unauthenticated-requests` to true.

Delegated access to a manager

The following ACI can be used to allow an employee's manager to edit the value of the employee's `telephoneNumber` attribute. This ACI uses the `userattr` keyword with a bind type of `USERDN`, which indicates that the target entry's manager attribute must have a value equal to the DN of the authenticated user:

```
aci: (targetattr="telephoneNumber")
(version 3.0; acl "A manager can update telephone numbers of her direct
 reports";
allow (read,search,compare,write) userattr="manager#USERDN";)
```

Proxy authorization

The following ACIs can be used to allow the application "`cn=OnBehalf,ou=applications,dc=example,dc=com`" to use the proxied authorization v2 control to request that operations be performed using an alternate authorization identity. The application user is also required to have the `proxied-auth` privilege as discussed later in this chapter:

```
aci: (version 3.0;acl "Application OnBehalf can proxy as another entry";
allow (proxy) userdn="ldap:///cn=OnBehalf,ou=applications,dc=example,dc=com";)
```

Validating ACIs before migrating data

Many directory servers allow for less restrictive application of their access control instructions, so that they accept invalid ACIs. For example, if Sun/Oracle encounters an access control rule that it cannot parse, then it will simply ignore it without any warning, and the server may not offer the intended access protection. Rather than unexpectedly exposing sensitive data, the PingDirectoryProxy Server rejects any ACIs that it cannot interpret, which ensures data access is properly limited as intended, but it can cause problems when migrating data with existing access control rules to a PingDirectoryProxy Server.

To validate an access control instruction, the PingDirectoryProxy Server provides a **`validate-acis`** tool in the `bin` directory (UNIX or Linux systems) or `bat` directory (Windows systems) that identifies any ACI syntax problems before migrating data. The tool can examine access control rules contained in either an LDIF file or an LDAP directory and write its result in LDIF with comments providing information about any problems that were identified. Each entry in the output will contain only a single ACI, so if an entry in the input contains multiple ACIs, then it may be present multiple times in the output, each time with a different ACI value. The entries contained in the output contains only ACI values, and all other attributes will be ignored.

Validating ACIs from a file

About this task

The **`validate-acis`** tool can process data contained in an LDIF file. It will ignore all attributes except `aci`, and will ignore all entries that do not contain the `aci` attribute, so any existing LDIF file that contains access control rules may be used.

Steps

1. Run the **`bin/validate-acis`** tool (UNIX or Linux systems) or **`bat\validate-acis`** (Win dows systems) by specifying the input file and output file. If the output file already exists, the existing contents will be re-written. If no output file is specified, then the results will be written to standard output.

```
$ bin/validate-acis --ldifFile test-acis.ldif --outputFile validated-
acis.ldif
```

```
# Processing complete # Total entries examined: 1
# Entries found with ACIs: 1
# Total ACI values found: 3
# Malformed ACI values found: 0
# Other processing errors encountered: 0
```

2. Review the results by opening the output file. For example, the `validated-acis.ldif` file that was generated in the previous step reads as follows:

```
# The following access control rule is valid
dn: dc=example,dc=com
aci: (targetattr!="userPassword")
  (version 3.0; acl "Allow anonymous read access for anyone";
    allow (read,search,compare) userdn="ldap:///anyone";)

# The following access control rule is valid
dn: dc=example,dc=com
aci: (targetattr="*")
  (version 3.0; acl "Allow users to update their own entries";
    allow (write) userdn="ldap:///self";)

# The following access control rule is valid
dn: dc=example,dc=com
aci: (targetattr="*")
  (version 3.0; acl "Grant full access for the admin user";
    allow (all) userdn="ldap:///uid=admin,dc=example,dc=com";)
```

3. If the input file has any malformed ACIs, then the generated output file will show what was incorrectly entered. For example, remove the quotation marks around `userPassword` in the original `test-acis.ldif` file, and re-run the command. The following command uses the `--onlyReportErrors` option to write any error messages to the output file only if a malformed ACI syntax is encountered.

```
$ bin/validate-acis --ldifFIle test-acis.ldif --outputFile validated-
acis.ldif \
  --onlyReportErrors
```

```
# Processing complete
# Total entries examined: 1
# Entries found with ACIs: 1
# Total ACI values found: 3
# Malformed ACI values found: 0
# Other processing errors encountered: 0
```

The output file shows the following message:

```
# The following access control rule is malformed or contains an unsupported
# syntax: The provided string '(targetattr!=userPassword)(version 3.0; acl
# "Allow anonymous read access for anyone"; allow (read,search,compare)
# userdn="ldap:///anyone";)' could not be parsed as a valid Access Control
# Instruction (ACI) because it failed general ACI syntax evaluation
dn: dc=example,dc=com
aci: (targetattr!=userPassword)
  (version 3.0; acl "Allow anonymous read access for anyone";
    allow (read,search,compare) userdn="ldap:///anyone";)
```

```
# The following access control rule is valid
dn: dc=example,dc=com
aci: (targetattr="*")
  (version 3.0; acl "Allow users to update their own entries";
    allow (write) userdn="ldap:///self";)

# The following access control rule is valid
dn: dc=example,dc=com
aci: (targetattr="*")
  (version 3.0; acl "Grant full access for the admin user";
    allow (all) userdn="ldap:///uid=admin,dc=example,dc=com";)
```

Validating ACIs in another Directory Proxy Server

About this task

The **`validate-acis`** tool also provides the ability to examine ACIs in data that exists in another Directory Proxy Server that you are planning to migrate to the PingDirectoryProxy Server. The tool helps to determine whether the Ping Identity Server accepts those ACIs.

Steps

- To use it in this manner, provide arguments that specify the address and port of the target Directory Proxy Server, credentials to use to bind, and the base DN of the subtree containing the ACIs to validate.

```
$ bin/validate-acis
```

```
# Processing complete # Total entries examined: 1
# Entries found with ACIs: 1
# Total ACI values found: 3
# Malformed ACI values found: 0
# Other processing errors encountered: 0
```

Migrating ACIs from Sun/Oracle to PingDirectory Server

This section describes the most important differences in access control evaluation between Sun/Oracle and the PingDirectory Server.

Support for macro ACIs

Sun/Oracle provides support for macros ACIs, making it possible to define a single ACI that can be used to apply the same access restrictions to multiple branches in the same basic structure. Macros ACIs are infrequently used and can cause severe performance degradation, so support for macros ACIs is not included in the PingDirectory Server. However, you can achieve the same result by simply creating the same ACIs in each branch.

Support for the roleDN bind rule

Sun/Oracle roles are a proprietary, non-standard grouping mechanism that provide little value over standard grouping mechanisms. The PingDirectory Server does not support DSEE roles and does not support the use of the `roleDN` ACI bind rule. However, the same behavior can be achieved by converting the DSEE roles to standard groups and using the `groupDN` ACI bind rule.

Targeting operational attributes

The Sun/Oracle access control model does not differentiate between user attributes and operational attributes. With Sun/Oracle, using `targetattr="*"` will automatically target both user and operational attributes. Using an exclusion list like `targetattr!="userPassword"` will automatically target all operational attributes in addition to all user attributes except `userPassword`. This behavior is responsible

for several significant security holes in which users are unintentionally given access to operational attributes. In some cases, it allows users to do things like exempt themselves from password policy restrictions.

In the PingDirectory Server, operational attributes are treated differently from user attributes and operational attributes are never automatically included. As such, `targetattr="*"` will target all user attributes but no operational attributes, and `targetattr!="userPassword"` will target all users attributes except `userPassword`, but no operational attributes. Specific operational attributes can be targeted by including the names in the list, like `targetattr="creatorsName||modifiersName"`. All operational attributes can be targeted using the `"+"` character. So, `targetattr="+"` targets all operational attributes but no user attributes and `targetattr="*||+"` targets all user and operational attributes.

Specification of global ACIs

Both DSEE and PingDirectory Server support global ACIs, which can be used to define ACIs that apply throughout the server. In servers with multiple naming contexts, this feature allows you to define a rule once as a global ACI, rather than needing to maintain an identical rule in each naming context.

In DSEE, global ACIs are created by modifying the root DSE entry to add values of the `aci` attribute. In the PingDirectory Server, global ACIs are managed with **`dsconfig`** referenced in the `global-aci` property of the Access Control Handler.

Defining ACIs for non-user content

In DSEE, you can write to the configuration, monitor, changelog, and tasks backends to define ACIs. In the PingDirectory Server, access control for private backends, like configuration, monitor, schema, changelog, tasks, encryption settings, backups, and alerts, should be defined as global ACIs.

Limiting access to controls and extended operations

DSEE offers limited support for restricting access to controls and extended operations. To the extent that it is possible to control such access with ACIs, DSEE defines entries with a DN such as "`oid={oid},cn=features,cn=config`" where `{oid}` is the OID of the associated control or extended operation. For example, the following DSEE entry defines ACIs for the persistent search control: "`oid=2.16.840.1.113730.3.4.3,cn=features,cn=config`".

In the PingDirectory Server, the "`targetcontrol`" keyword can be used to define ACIs that grant or deny access to controls. The "`extop`" keyword can be used to define ACIs that grant or deny access to extended operation requests.

Tolerance for malformed ACI values

In DSEE, if the server encounters a malformed access control rule, it simply ignores that rule without any warning. If this occurs, then the server will be running with less than the intended set of ACIs, which may prevent access to data that should have been allowed or, worse yet, may grant access to data that should have been restricted.

The PingDirectory Server is much more strict about the access control rules that it will accept. When performing an LDIF import, any entry containing a malformed or unsupported access control rule will be rejected. Similarly, any add or modify request that attempts to create an invalid ACI will be rejected. In the unlikely event that a malformed ACI does make it into the data, then the server immediately places itself in lockdown mode, in which the server terminates connections and rejects requests from users without the `lockdown-mode` privilege. Lockdown mode allows an administrator to correct the problem without risking exposure to user data.

ⓘ **Note:** Consider running the **`import-ldif`** tool with the `--rejectFile` option so that you can review any rejected ACIs.

About the privilege subsystem

In DSEE, only the root user is exempt from access control evaluation. While administrators can create ACIs that give "normal" users full access to any content, they can also create ACIs that would make some portion of the data inaccessible even to those users. In addition, some tasks can only be accomplished by the root user and you cannot restrict the capabilities assigned to that root user.

The PingDirectory Server offers a privilege subsystem that makes it possible to control the capabilities available to various users. Non-root users can be granted limited access to certain administrative capabilities, and restrictions can be enforced on root users. In addition, certain particularly risky actions (such as the ability to interact with the server configuration, change another user's password, impersonate another user, or shutdown and restart the server) require that the requester have certain privileges in addition to sufficient access control rights to process the operation.

Identifying unsupported ACIs

The PingDirectory Server provides a **`validate-acis`** tool that can be used to examine content in an LDIF file or data in another directory server (such as a DSEE instance) to determine whether the access control rules contained in that data are suitable for use in the PingDirectory Server instance. When migrating data from a DSEE deployment into a PingDirectory Server instance, the **`validate-acis`** tool should first be used to determine whether ACIs contained in the data are acceptable. If any problems are identified, then the data should be updated to correct or redefine the ACIs so that they are suitable for use in the PingDirectory Server.

For more information about using this tool, see *Validating ACIs Before Migrating Data*.

Working with privileges

In addition to the access control implementation, the PingDirectoryProxy Server includes a privilege subsystem that can also be used to control what users are allowed to do. The privilege subsystem works in conjunction with the access control subsystem so that privileged operations are only allowed if they are allowed by the access control configuration and the user has all of the necessary privileges.

Privileges can be used to grant normal users the ability to perform certain tasks that, in most other directories, would only be allowed for the root user. In fact, the capabilities extended to root users in the PingDirectoryProxy Server are all granted through privileges, so you can create a normal user account with the ability to perform some or all of the same actions as root users.

Administrators can also remove privileges from root users so that they are unable to perform certain types of operations. Multiple root users can be defined in the server with different sets of privileges so that the capabilities that they have are restricted to only the tasks that they need to be able to perform.

Available privileges

The following privileges are defined in the PingDirectoryProxy Server.

Summary of Privileges

Privilege	Description
audit-data-security	This privilege is required to initiate a data security audit on the server, which is invoked by the **`audit-data-security`** tool.
backend-backup	This privilege is required to initiate an online backup through the tasks interface. The server's access control configuration must also allow the user to add the corresponding entry in the tasks backend.
backend-restore	This privilege is required to initiate an online restore through the tasks interface. The server's access control configuration must also allow the user to add the corresponding entry in the tasks backend.

Privilege	Description
bypass-acl	This privilege allows a user to bypass access control evaluation. For a user with this privilege, any access control determination made by the server immediately returns that the operation is allowed. Note, however, that this does not bypass privilege evaluation, so the user must have the appropriate set of additional privileges to be able to perform any privileged operation (for example, a user with the `bypass-acl` privilege but without the `config-read` privilege is not allowed to access the server configuration).
bypass-pw-policy	This privilege allows a user entry to bypass password policy evaluation. This privilege is intended for cases where external synchronization might require passwords that violate the password validation rules. The privilege is also evaluated for bind operations, meaning password restrictions are also bypassed when binding as a user who has this privilege.
bypass-read-acl	This privilege allows the associated user to bypass access control checks performed by the server for bind, search, and compare operations. Access control evaluation may still be enforced for other types of operations.
config-read	This privilege is required for a user to access the server configuration. Access control evaluation is still performed and can be used to restrict the set of configuration objects that the user is allowed to see.
config-write	This privilege is required for a user to alter the server configuration. The user is also required to have the `config-read` privilege. Access control evaluation is still performed and can be used to restrict the set of configuration objects that the user is allowed to alter.
disconnect-client	This privilege is required for a user to request that an existing client connection be terminated. The connection is terminated through the disconnect client task. The server's access control configuration must also allow the user to add the corresponding entry to the tasks backend.
jmx-notify	This privilege is required for a user to subscribe to JMX notifications generated by the Directory Proxy Server. The user is also required to have the `jmx-read` privilege.
jmx-read	This privilege is required for a user to access any information provided by the Directory Proxy Server via the Java Management Extensions (JMX).
jmx-write	This privilege is required for a user to update any information exposed by the Directory Proxy Server via the Java Management Extensions (JMX). The user is also required to have the `jmx-read` privilege. Note that currently all of the information exposed by the server over JMX is read-only.
ldif-export	This privilege is required to initiate an online LDIF export through the tasks interface. The server's access control configuration must also allow the user to add the corresponding entry in the Tasks backend. To allow access to the Tasks backend, you can set up a global ACI that allows access to members of an Administrators group.
ldif-import	This privilege is required to initiate an online LDIF import through the tasks interface. The server's access control configuration must also allow the user to add the corresponding entry in the Tasks backend. To allow access to the Tasks backend, configure the global ACI as shown in the previous description of the **`ldif-export`** privilege.
lockdown-mode	This privilege allows the associated user to request that the server enter or leave lockdown mode, or to perform operations while the server is in lockdown mode.

Privilege	Description
modify-acl	This privilege is required for a user to add, modify, or remove access control rules defined in the server. The server's access control configuration must also allow the user to make the corresponding change to the `aci` operational attribute.
password-reset	This privilege is required for one user to be allowed to change another user's password. This privilege is not required for a user to be allowed to change his or her own password. The user must also have the access control instruction privilege to write the `userPassword` attribute to the target entry.
privilege-change	This privilege is required for a user to change the set of privileges assigned to a user, including the set of privileges, which are automatically granted to root users. The server's access control configuration must also allow the user to make the corresponding change to the `ds-privilege-name` operational attribute.
proxied-auth	This privilege is required for a user to request that an operation be performed with an alternate authorization identity. This privilege applies to operations that include the proxied authorization v1 or v2 control operations that include the intermediate client request control with a value set for the client identity field, or for SASL bind requests that can include an authorization identity different from the authentication identity.
server-restart	This privilege is required to initiate a server restart through the tasks interface. The server's access control configuration must also allow the user to add the corresponding entry in the tasks backend.
server-shutdown	This privilege is required to initiate a server shutdown through the tasks interface. The server's access control configuration must also allow the user to add the corresponding entry in the tasks backend.
soft-delete-read	This privilege is required for a user to access a soft-deleted-entry.
stream-values	This privilege is required for a user to perform a stream values extended operation, which obtains all entry DNs and/or all values for one or more attributes for a specified portion of the DIT.
unindexed-search	This privilege is required for a user to be able to perform a search operation in which a reasonable set of candidate entries cannot be determined using the defined index and instead, a significant portion of the database needs to be traversed to identify matching entries. The server's access control configuration must also allow the user to request the search.
update-schema	This privilege is required for a user to modify the server schema. The server's access control configuration must allow the user to update the operational attributes that contain the schema elements.

Privileges automatically granted to root users

The special abilities that root users have are granted through privileges. Privileges can be assigned to root users in two ways:

- By default, root users may be granted a specified set of privileges. Note that it is possible to create root users which are not automatically granted these privileges by including the `ds-cfg-inherit-default-root-privileges` attribute with a value of FALSE in the entries for those root users.
- Individual root users can have additional privileges granted to them, and/or some automatically-granted privileges may be removed from that user.

The set of privileges that are automatically granted to root users is controlled by the `default-root-privilege-name` property of the Root DN configuration object. By default, this set of privileges includes:

- audit-data-security
- backend-backup
- backend-restore
- bypass-acl
- config-read
- config-write
- disconnect-client
- ldif-export
- lockdown-mode
- manage-topology
- metrics-read
- modify-acl
- password-reset
- permit-get-password-policy-state-issues
- privilege-change
- server-restart
- server-shutdown
- soft-delete-read
- stream-values
- unindexed-search
- update-schema

The privileges not granted to root users by default includes:

- bypass-pw-policy
- bypass-read-acl
- jmx-read
- jmx-write
- jmx-notify
- permit-externally-processed-authentication
- permit-proxied-mschapv2-details
- proxied-auth

The set of default root privileges can be altered to add or remove values as necessary. Doing so will require the `config-read`, `config-write`, and `privilege-change` privileges, as well as either the `bypass-acl` privilege or sufficient permission granted by the access control configuration to make the change to the server's configuration.

Assigning additional privileges for administrators

To allow access to the Tasks backend, set up a global ACI that allows access to members of an Administrators group as follows:

```
$ dsconfig set-access-control-handler-prop \
  --add 'global-aci:(target="ldap:///cn=tasks")(targetattr="*||+")
        (version 5.0; acl "Access to the tasks backend for administrators";
         allow (all) groupdn="ldap:///
         cn=admins,ou=groups,dc=example,dc=com";)'
```

Assigning privileges to normal users and individual root users

Privileges can be granted to normal users on an individual basis. This can be accomplished by adding the `ds-privilege-name` operational attribute to that user's entry with the names of the desired privileges. For example, the following change will grant the `proxied-auth` privilege to the `uid=proxy,dc=example,dc=com` account:

```
dn: uid=proxy,dc=example,dc=com
changetype: modify
add: ds-privilege-name
ds-privilege-name: proxied-auth
```

The user making this change will be required to have the `privilege-change` privilege, and the server's access control configuration must also allow the requester to write to the `ds-privilege-name` attribute in the target user's entry.

This same method can be used to grant privileges to root users that they would not otherwise have through the set of default root privileges. You can also remove default root privileges from root users by prefixing the name of the privilege to remove with a minus sign. For example, the following change grants a root user the `jmx-read` privilege in addition to the set of default root privileges, and removes the `server-restart` and `server-shutdown` privileges:

```
dn: cn=Sync Root User,cn=Root DNs,cn=config
changetype: modify
add: ds-privilege-name
ds-privilege-name: jmx-read
ds-privilege-name: -server-restart
ds-privilege-name: -server-shutdown
```

Note that because root user entries exist in the configuration, this update requires the `config-read` and `config-write` privileges in addition to the `privilege-change` privilege.

Disabling privileges

Although the privilege subsystem in the PingDirectoryProxy Server is a very powerful feature, it might break some applications if they expect to perform some operation that requires a privilege that they do not have. In the vast majority of these cases, you can work around the problem by simply assigning the necessary privilege manually to the account used by that application. However, if this workaround is not sufficient, or if you need to remove a particular privilege (for example, to allow anyone to access information via JMX without requiring the `jmx-read` privilege), then privileges can be disabled on an individual basis.

The set of disabled privileges is controlled by the `disabled-privilege` property in the global configuration object. By default, no privileges are disabled. If a privilege is disabled, then the server behaves as if all users have that privilege.

Deploying a Standard Directory Proxy Server

You can deploy Directory Proxy Server in a variety of ways, depending upon the needs of your enterprise. This chapter describes and illustrates a standard deployment scenario.

Introduction

Derived from the words *development* and *operations*, the term *DevOps* refers to the practices that a company follows to ensure the production of high-quality products, while also minimizing the amount of time between the commitment of a system change and the implementation of that change in a production environment.

Most companies practice one of the following service models:

- Pets — With the *pets service model*, servers are built and managed manually, and are treated as unique and indispensable. Examples include mainframes, database systems, and load balancers.
- Cattle — With the *cattle service model*, arrays of multiple replaceable servers are built with automated tools. During a failure event, an array automatically restarts failed services and replicates data. Examples include web server arrays, search clusters, and multi-master datastores.

Historically, servers have been treated like pets. The failure of one or multiple servers was often viewed as an emergency, and extensive resources were usually required to repair the damage. In the new DevOps paradigm, servers are recognized as dispensable and treated like cattle. When a server fails, the infrastructure replaces it immediately, configuring the replacement server identically to the failed one. Because no human intervention is required to fix them, the servers are considered self-healing.

To help treat your servers more like cattle than pets, PingDirectoryProxy Server supports server profiles and features like topology-management tools. For example, the **`manage-profile setup`** represents a single command that performs all the steps from the pet service model on a `server-profile` directory, a well-defined directory structure with all the necessary server configuration bits. Similarly, the **`manage-profile replace-profile`** command performs similar steps with a single command invocation after a server is updated to a new version.

The scripts in the `server-profile` directory are declarative of the environment. Consequently, what you define in the `server-profile` directory is what you get on the servers. No one needs to identify a server's current configuration and compute the differences that must be applied to attain the appropriate end state. Prior to server profiles, this problem was difficult to address, especially where no history was available. In such scenarios, an administrator might have needed to obtain the current configuration from the servers, to manually compute the difference between the current and desired configuration, and to apply the configuration changes, hoping all the while that nobody had changed the configuration during the process. Server profiles eliminate this procedural approach to applying configuration changes, and they simplify the steps associated with determining what is deployed in an environment. For more information on server profiles, see *Server profiles* on page 816.

Another principle that relates closely to DevOps is infrastructure as code (IaC), the concept of managing your operations in the same manner as your application and other code for general release with proper versioning, continuous integration, quality control, and release cycles. Customers who deploy PingDirectoryProxy Servers as pets today can take advantage of current DevOps and IaC principles to turn them into cattle.

Automatic server discovery

Version 8.1 supports the use of information in the topology registry to automatically discover the backend Directory Server instances to which requests may be forwarded. This is an alternative to explicitly configuring the PingDirectoryProxy Server with the backend servers that should be used. This option is only available if all of the backend servers are Directory Server instances, and only if the following are true:

- The PingDirectoryProxy Server must be a member of the same topology as the Directory Server instances to which requests will be forwarded.
- The PingDirectoryProxy Server must be configured with an LDAP external server template that provides the settings to use when communicating with dynamically discovered backend Directory Server instances.
- The PingDirectoryProxy Server's load-balancing algorithms must be configured to use the desired LDAP external server template.
- The topology registry entries for each Directory Server instance must be updated to indicate which of the PingDirectoryProxy Server's load-balancing algorithms can forward requests to the server.

ⓘ **Note:** When using automatic server discovery, you do not need to run the **`prepare-external-server`** command. Servers do not need a service account and associated ACIs before discovery since the inter-server bind process is used to initiate communication. The server is identified using information contained in the topology registry and is given appropriate privileges during the inter-server bind process.

Joining a PingDirectoryProxy Server to an existing PingDirectory Server topology

PingDirectory Server Version 8.0.0.0 supports the addition of PingDirectoryProxy Server instances to the same topology as Directory Server instances. This can be done when the PingDirectoryProxy Server instance is initially configured, using either the setup utility (in either interactive or non-interactive mode)

or the manage-profile setup command. This can also be done later using the manage-topology add-server command.

Joining a topology with interactive setup

If you run **`setup`** without any arguments, it starts in interactive mode and prompts you for all of the necessary information. After you accept the license, the next prompt asks if you want to add the server to an existing PingDirectoryProxy Server topology. If you have at least one PingDirectoryProxy Server instance that is already in the desired topology, you can enter “yes” at this prompt and it will walk you through the process of creating a new instance that is a copy of the existing instance (with all of the same configuration). You are then asked for the information needed to connect and authenticate to the existing PingDirectoryProxy Server instance. For example:

```
Do you accept the terms of this license agreement?  Enter 'yes' to accept,
'no' to reject, or press ENTER to display the next page of the agreement []:
 yes

Would you like to add this server to an existing Directory Proxy Server
topology? (yes / no) [no]: yes

Enter the host name of the peer Directory Proxy Server from which you would
like to copy configuration settings. [proxy2.example.com]:
 proxy1.example.com

Enter the LDAP port of the peer Directory Proxy Server [389]: 636

How would you like to connect to the peer Directory Proxy Server?

    1)  None
    2)  SSL
    3)  StartTLS

Enter option [1]: 2

Enter the manager account DN for the peer Directory Proxy Server
 [cn=Directory
Manager]: cn=Directory Manager

Enter the password for cn=Directory Manager:
The server presented the following certificate chain:

     Subject: CN=proxy1.example.com,O=Example Corp,C=US
     Valid From: Saturday, November 2, 2019 at 10:34:09 PM CDT
     Valid Until: Sunday, November 1, 2020 at 09:34:09 PM CST
     SHA-1 Fingerprint:
 54:7f:6c:c1:99:73:c4:19:66:6e:da:4b:ee:a9:d5:62:24:2e:ba:41
     256-bit SHA-2 Fingerprint:
 54:ce:59:c5:25:85:95:17:17:69:e0:5c:57:9e:ed:27:3d:af:9c:bd:34:51:c8:46:1e:e4:2f:31:13
     -
     Issuer 1 Subject: CN=Example Certification Authority,O=Example
 Corp,C=US
     Valid From: Saturday, November 2, 2019 at 10:34:03 PM CDT
     Valid Until: Friday, October 28, 2039 at 10:34:03 PM CDT
     SHA-1 Fingerprint:
 34:25:1f:8f:18:ff:a8:a9:ac:22:d3:d2:fc:bb:0b:4c:53:e1:8c:de
     256-bit SHA-2 Fingerprint:
 51:69:1f:bb:cf:6f:1c:7a:e6:d4:6d:5a:01:c7:08:45:88:53:fc:75:f1:63:bb:ec:65:f1:1f:4e:26

Do you wish to trust this certificate?  Enter 'y' or 'n': y


Initializing ..... Done
Reading Peer Configuration ..... Done
```

```
Connecting to 'proxy1' ..... Done
```

However, cloning an existing installation isn't possible when setting up the first PingDirectoryProxy Server instance. In this case, if you enter "no" at the prompt to join an existing PingDirectoryProxy Server topology, setup asks if you want to join a Directory Server topology instead. If you enter "yes", the process is basically the same as joining an existing PingDirectoryProxy Server topology and you are prompted for the information needed to connect and authenticate to a Directory Server instance in the topology. The primary difference is that you have to define the PingDirectoryProxy Server configuration yourself. For example:

```
Do you accept the terms of this license agreement?  Enter 'yes' to accept,
'no' to reject, or press ENTER to display the next page of the agreement []:
 yes

Would you like to add this server to an existing Directory Proxy Server
topology? (yes / no) [no]: no

Would you like to add this server to an existing PingDirectory server
 topology
to enable automatic backend server discovery? (yes / no) [no]: yes

Enter the host name of a PingDirectory server instance in the topology to
join. [proxy1.example.com]: ds1.example.com

Enter the LDAP port of a PingDirectory server instance in the topology to
 join
[389]: 636

How would you like to secure communication with the PingDirectory server

    1)  None
    2)  SSL
    3)  StartTLS

Enter option [1]: 2

Enter the DN used to bind to a PingDirectory server instance in the topology
to join [cn=Directory Manager]: cn=Directory Manager

Enter the password for cn=Directory Manager:
Testing connection to the existing PingDirectory server topology
The server presented the following certificate chain:

     Subject: CN=ds1.example.com,O=Example Corp,C=US
     Valid From: Saturday, November 2, 2019 at 10:44:26 PM CDT
     Valid Until: Sunday, November 1, 2020 at 09:44:26 PM CST
     SHA-1 Fingerprint:
 e1:4a:9e:dc:55:e8:40:78:9b:e1:1b:bd:3e:4c:85:fb:60:b4:27:35
     256-bit SHA-2 Fingerprint:
 6e:92:c7:d6:66:c8:3d:2d:04:4c:f2:6a:cb:cb:51:5a:bf:f8:d6:18:0a:fc:64:d9:76:f4:4e:58:eb
     -
     Issuer 1 Subject: CN=Example Certification Authority,O=Example
 Corp,C=US
     Valid From: Saturday, November 2, 2019 at 10:44:23 PM CDT
     Valid Until: Friday, October 28, 2039 at 10:44:23 PM CDT
     SHA-1 Fingerprint:
 9a:b7:aa:a3:33:49:ce:b8:f3:7e:60:13:e0:3c:63:4b:8f:95:7a:f3
     256-bit SHA-2 Fingerprint:
 04:07:86:f2:5c:e2:c1:88:fe:08:27:c1:1e:52:b0:4b:98:6e:a8:5c:85:fc:e0:d9:25:4f:07:ae:d7

Do you wish to trust this certificate?  Enter 'y' or 'n': y
Successfully connected to the existing PingDirectory server topology
```

Joining a topology with non-interactive setup

Interactive mode is a convenient method to get the server up and running, especially when you are just getting started, but the installation process for production deployments is generally scripted. For this process, non-interactive mode is a better choice and setup offers a number of useful arguments.

When creating the initial Directory Proxy Server instance, you can use the following arguments to join an existing Directory Server topology:

- --existingDSTopologyHostName *{address}* - The address of a Directory Server instance in the topology to be joined.
- --existingDSTopologyPort *{port}* - The port for communication with the Directory Server to retrieve information about the topology.
- --existingDSTopologyUseSSL - Indicates that the communication with the Directory Server to retrieve information about the topology should be encrypted with SSL.
- --existingDSTopologyUseStartTLS - Indicates that the communication with the Directory Server to retrieve information about the topology should be encrypted with the StartTLS extended operation.
- --existingDSTopologyUseNoSecurity - Indicates that the communication with the Directory Server to retrieve information about the topology should be not be encrypted.
- --existingDSTopologyUseJavaTruststore *{path}* - The path to a JKS trust store that has the information needed to trust the certificate presented by the Directory Server when using SSL or StartTLS.
- --existingDSTopologyUsePkcs12Truststore *{path}* - The path to a PKCS #12 trust store that has the information needed to trust the certificate presented by the Directory Server when using SSL or StartTLS.
- --existingDSTopologyTrustStorePassword *{password}* - The password needed to access the contents of the JKS or PKCS #12 trust store. A password is typically required when using a PKCS #12 trust store but is optional when using a JKS trust store.
- --existingDSTopologyTrustStorePasswordFile *{path}* - The path to a file containing the password needed to access the contents of the JKS or PKCS #12 trust store.
- --existingDSTopologyBindDN *{path}* - The DN of the account to use to authenticate to the Directory Server. This account must have full read and write access to the configuration and to manage the topology.
- --existingDSTopologyBindPassword *{password}* - The password for the account to use to authenticate to the Directory Server.
- --existingDSTopologyBindPasswordFile *{path}* - The path to a file containing the password to use to authenticate to the Directory Server.

For example, you can use a command similar to the following to set up a PingDirectoryProxy Server instance in the same topology as a Directory Server instance:

```
$ ./setup --acceptLicense \
     --licenseKeyFile PingDirectory.lic
     --maxHeapSize 2g \
     --localHostName proxy1.example.com \
     --skipHostnameCheck \
     --instanceName proxy1 \
     --location Austin \
     --rootUserDN "cn=Directory Manager" \
     --rootUserPasswordFile directory-manager-password.txt \
     --ldapPort 389 \
     --ldapsPort 636 \
     --httpsPort 443 \
     --enableStartTLS \
     --useJavaKeyStore config/keystore \
     --keyStorePasswordFile config/keystore.pin \
     --certNickname server-cert \
     --useJavaTrustStore config/truststore \
     --trustStorePasswordFile config/truststore.pin \
     --encryptDataWithPassphraseFromFile encryption-passphrase.txt \
```

```
      --existingDSTopologyHostName ds1.example.com \
      --existingDSTopologyPort 636 \
      --existingDSTopologyBindDN "cn=Directory Manager" \
      --existingDSTopologyBindPasswordFile directory-manager-password.txt \
      --existingDSTopologyUseSSL \
      --existingDSTopologyUseJavaTrustStore config/truststore \
      --no-prompt
```

As with interactive mode, it is possible to use non-interactive mode to clone the configuration of an existing PingDirectoryProxy Server instance, including joining the same topology as the existing instance. You can use the following arguments to do this rather than the arguments listed above:

- --peerHostName *{address}* - The address of a PingDirectoryProxy Server instance whose configuration should be cloned and whose topology should be joined.
- --peerPort *{port}* - The port communication with the PingDirectoryProxy Server to retrieve the configuration and topology information.
- --peerUseSSL - Indicates that communication with the PingDirectoryProxy Server to retrieve configuration and topology information should be encrypted with SSL.
- --peerUseStartTLS - Indicates that communication with the PingDirectoryProxy Server to retrieve configuration and topology information should be encrypted with the StartTLS extended operation.
- --peerUseNoSecurity - Indicates that communication with the PingDirectoryProxy Server to retrieve configuration and topology information should not be encrypted.

ⓘ **Note:** When using SSL or StartTLS to encrypt the communication, you also need to use one of the --useJavaTruststore or --usePkcs12Truststore arguments to specify the path to a trust store with the information needed to trust the certificate that is presented by the PingDirectoryProxy Server.

The following is an example of a sample command to set up a new PingDirectoryProxy Server as a clone of an existing PingDirectoryProxy Server instance:

```
$ ./setup --acceptLicense \
      --licenseKeyFile PingDirectory.lic
      --maxHeapSize 2g \
      --localHostName proxy2.example.com \
      --skipHostnameCheck \
      --instanceName proxy2 \
      --location Austin \
      --rootUserDN "cn=Directory Manager" \
      --rootUserPasswordFile directory-manager-password.txt \
      --ldapPort 389 \
      --ldapsPort 636 \
      --httpsPort 443 \
      --enableStartTLS \
      --useJavaKeyStore config/keystore \
      --keyStorePasswordFile config/keystore.pin \
      --certNickname server-cert \
      --useJavaTrustStore config/truststore \
      --trustStorePasswordFile config/truststore.pin \
      --encryptDataWithPassphraseFromFile encryption-passphrase.txt \
      --peerHostName proxy1.example.com \
      --peerPort 636 \
      --peerUseSSL \
      --no-prompt
```

Joining a topology with manage-profile setup

The **`manage-profile`** tool can be used to set up an instance of the server from information contained in a server profile. This is like an enhanced version of the **`setup`** utility in non-interactive mode because not only does it invoke **`setup`**, but it can also perform other tasks like applying configuration changes, installing schema and extensions, and adding files to the server root.

Since **manage-profile setup** uses the setup tool in non-interactive mode, you should use the arguments listed in the previous section, including the following:

- --existingDSTopologyHostName *{address}*
- --existingDSTopologyPort *{port}*
- --existingDSTopologyUseSSL
- --existingDSTopologyUseStartTLS
- --existingDSTopologyUseNoSecurity
- --existingDSTopologyUseJavaTruststore *{path}*
- --existingDSTopologyUsePkcs12Truststore *{path}*
- --existingDSTopologyTrustStorePassword *{password}*
- --existingDSTopologyTrustStorePasswordFile *{path}*
- --existingDSTopologyBindDN {path}
- --existingDSTopologyBindPassword *{password}*
- --existingDSTopologyBindPasswordFile *{path}*

An appropriate set of arguments should be placed in the `setup-arguments.txt` file in the root directory of the profile, along with all of the other arguments that should be used when invoking **setup**.

If you have already set up an instance of the server, you can run **manage-profile generate-profile** to generate a profile from the information contained in that instance. If the server was added to the topology during the setup process, the generated profile will include an appropriate set of arguments for joining the same topology.

Note that unlike the **setup** utility, **manage-profile setup** does not support cloning an existing PingDirectoryProxy Server instance, so the --peerHostName, --peerPort, and other related arguments cannot be included in the `setup-arguments.txt` file.

Joining a topology with manage-topology add-server

You can use the **manage-topology add-server** command to add a PingDirectoryProxy Server instance to a topology after it has been installed. You can only do this if the PingDirectoryProxy Server instance is not already part of any other topology, since it is not possible to join two topologies together.

This tool supports all of the normal arguments for connecting and authenticating to the local server instance, including the following:

- --hostname *{address}*
- --port *{port}*
- --useSSL
- --useStartTLS
- --trustStorePath *{path}*
- --trustStorePassword *{password}*
- --trustStorePasswordFile *{path}*
- --bindDN *{dn}*
- --bindPassword *{password}*
- --bindPasswordFile *{path}*

The **manage-topology add-server** command also allows the following arguments to provide information about a server in the topology to be joined:

- --remoteServerHostname *{address}* - The address of a server in the topology to be joined.
- --remoteServerPort *{port}* - The port for communication with the remote server.
- --remoteServerConnectionSecurity {noSecurity|useSSL|useStartTLS} - The type of security to use when communicating with the remote server. This value must be one of useSSL (to indicate that the communication should be encrypted with SSL), useStartTLS (to indicate that the communication should be encrypted with the StartTLS extended operation), or noSecurity (to indicate that the communication should not be encrypted).

- --remoteServerBindDN *{dn}* - The DN of the account to use to authenticate to the remote server.
- --remoteServerBindPassword *{password}* - The password for the account to use to authenticate to the remote server.
- --remoteServerBindPasswordFile *{path}* - The path to a file containing the password for the account to use to authenticate to the remote server.

For example, a command similar to the following could be used to add a PingDirectoryProxy Server to an existing Directory Server topology:

```
$ bin/manage-topology add-server \
     --hostname proxy1.example.com \
     --port 636 \
     --useSSL \
     --trustStorePath config/truststore \
     --bindDN "cn=Directory Manager" \
     --bindPasswordFile directory-manager-password.txt \
     --remoteServerHostname ds1.example.com \
     --remoteServerPort 636 \
     --remoteServerConnectionSecurity useSSL \
     --remoteServerBindDN "cn=Directory Manager" \
     --remoteServerBindPasswordFile directory-manager-password.txt
```

Creating an LDAP external server template

An LDAP external server template is a configuration object that can be used to provide a load-balancing algorithm with many of the settings that it should use when communicating with a backend server that has been discovered from the topology registry. An LDAP external server template configuration object has most of the same properties as an LDAP external server configuration object, but omits those related to information that it obtains from the topology registry. The omitted properties include:

- server-host-name
- server-port
- location
- connection-security

In addition, the health-check-state property is also not available for LDAP external server templates since it primarily applies to individual servers rather than all of the servers associated with a load-balancing algorithm.

Because the only LDAP servers which can be in the topology registry are Directory Servers, most of the remaining properties in LDAP external server templates have the same default values as the corresponding properties in the Directory Server external server type. However, there are a couple of exceptions, including the following:

- The authentication-method property has a default value of inter-server in LDAP external server templates, while it has a default value of simple in Directory Server external servers. The inter-server authentication type indicates that the PingDirectoryProxy Server should authenticate to the Directory Server with a proprietary authentication method that uses inter-server certificates stored in the topology registry.

  ⓘ **Note:** This option is only supported if all of the Directory Server instances are version 8.0.0.0 or later.
- The key-manager-provider property has a default value of Null in LDAP external server templates, while it has no default value in Directory Server external servers. When using the inter-server authentication type, the topology registry is used to obtain the inter-server certificates, so no additional key manager provider is required.
- The trust-manager-provider property has a default value of JVM-Default in LDAP external server templates, while it has no default value in Directory Server external servers. When using the inter-

server authentication type, the topology registry is used to obtain information about the listener certificates that the servers are expected to present.

In many cases the PingDirectoryProxy Server's default settings for an LDAP external server template are acceptable for most properties. However, you may wish to add custom health checks that will be invoked against servers created from the template. The PingDirectoryProxy Server will automatically check to see whether the server reports any degraded or unavailable alert types, and will also verify that the backend server's root DSE is accessible in a timely manner, but you may wish to add additional health checks, including the following:

- A search health check that verifies that the base entry from the associated subtree view can be retrieved in a timely manner.
- A replication backlog health check that verifies that replication is working and that none of the servers is too far out of sync.

The following example demonstrates the process for creating these health checks and then creating an LDAP external server template that uses them:

```
# Create a health check to verify that the dc=example,dc=com entry can be
# retrieved in a timely manner.
dsconfig create-ldap-health-check \
     --check-name dc_example_dc_com-retrieve-base-entry \
     --type search \
     --set enabled:true \
     --set base-dn:dc=example,dc=com \
     --set allow-no-entries-returned:false \
     --set allow-multiple-entries-returned:false

# Create a health check to verify that replication is working without a
# significant backlog.
dsconfig create-ldap-health-check \
     --check-name dc_example_dc_com-replication-backlog \
     --type replication-backlog \
     --set enabled:true \
     --set base-dn:dc=example,dc=com

# Create an LDAP external server template with the above
dsconfig create-ldap-external-server-template \
     --template-name dc_example_dc_com \
     --set health-check:dc_example_dc_com-retrieve-base-entry \
     --set health-check:dc_example_dc_com-replication-backlog
```

Defining the load-balancing algorithm configuration

In versions prior to 8.0.0.0, each load-balancing algorithm had to be explicitly configured with a set of backend servers. This is still supported, and may be the preferred configuration in some cases. For example, this is required if any backend server is not a Directory Server, or if it is a Directory Server prior to version 8.0.0.0. It may also be preferable if you wish to have more direct control over the configuration that the PingDirectoryProxy Server uses for each backend server.

Beginning with version 8.0.0.0, it is possible to configure the fewest operations and failover load-balancing algorithms so that they can automatically discover the appropriate set of backend servers from information in the topology registry. To do this, the load-balancing algorithm should be configured with a value for the ldap-external-server-template property rather than with one or more values for the backend-server property.

For example, the following configuration change can be used to create a fewest operations load-balancing algorithm that uses an LDAP external server template named dc_example_dc_com:

```
dsconfig create-load-balancing-algorithm \
     --algorithm-name fewest-operations-load-balancer \
     --type fewest-operations \
```

```
     --set enabled:true \
     --set ldap-external-server-template:dc_example_dc_com
```

You can use a similar change to create an instance of a failover load-balancing algorithm that will automatically discover its backend servers:

```
dsconfig create-load-balancing-algorithm \
     --algorithm-name failover-load-balancer \
     --type failover \
     --set enabled:true \
     --set ldap-external-server-template:dc_example_dc_com
```

Note that with the failover load-balancing algorithm, the backend servers will be ordered lexicographically by instance name. This ensures that all PingDirectoryProxy Server instances will have a consistent ordering, but it is not as easy to control the ordering when automatically discovering backend servers as when they are explicitly configured. This is acceptable in many cases, as the primary use of failover load-balancing is often to avoid replication and unique attribute conflicts for write operations, and potentially to avoid read-after-write issues. However, if you wish to direct all traffic to a specific instance as long as it is available, then it may be better to explicitly configure the set of backend servers rather than use automatic discovery.

Although you can use just the fewest operations load-balancing algorithm or the failover load-balancing algorithm, it is recommended that you use both of these together, with failover load-balancing for writes and fewest operations for reads. When combined with the Directory Server's default use of assured replication (for all add, delete, and modify DN operations, as well as for modify operations that target passwords or password-related attributes), this provides an excellent balance that minimizes the chance for replication and unique attribute conflicts, minimizes the chance for read-after-write issues that result from propagation delay, and maximizes performance and minimizes response time for read operations. You can do this by using criteria-based load balancing, which involves the following:

- Creating a failover load-balancing algorithm instance for write operations
- Creating a fewest operations load-balancing algorithm instance for read operations
- Creating a criteria-based load-balancing algorithm instance that uses failover load-balancing for write operations
- Configuring the proxying request processor to use the above criteria-based load balancing algorithm for writes and the fewest operations load-balancing algorithm for everything else

The following is an example of this configuration::

```
# Create a health check to verify that the dc=example,dc=com entry can be
# retrieved in a timely manner.
dsconfig create-ldap-health-check \
     --check-name dc_example_dc_com-retrieve-base-entry \
     --type search \
     --set enabled:true \
     --set base-dn:dc=example,dc=com \
     --set allow-no-entries-returned:false \
     --set allow-multiple-entries-returned:false

# Create a health check to verify that replication is working without a
# significant backlog.
dsconfig create-ldap-health-check \
     --check-name dc_example_dc_com-replication-backlog \
     --type replication-backlog \
     --set enabled:true \
     --set base-dn:dc=example,dc=com

# Create an LDAP external server template with the above
dsconfig create-ldap-external-server-template \
     --template-name dc_example_dc_com \
     --set health-check:dc_example_dc_com-retrieve-base-entry \
```

```
     --set health-check:dc_example_dc_com-replication-backlog

# Create a fewest operations failover load-balancing algorithm instance that
# uses the default LDAP external server template for backend servers that it
# discovers from the topology registry.
dsconfig create-load-balancing-algorithm \
     --algorithm-name dc_example_dc_com-fewest-operations \
     --type fewest-operations \
     --set enabled:true \
     --set ldap-external-server-template:dc_example_dc_com

# Create a failover load-balancing algorithm instance that also uses the
# default LDAP external server template for backend servers that it
# discovers from the topology registry.
dsconfig create-load-balancing-algorithm \
     --algorithm-name dc_example_dc_com-failover \
     --type failover \
     --set enabled:true \
     --set ldap-external-server-template:dc_example_dc_com

# Create a criteria-based load-balancing algorithm instance that will use
# failover load-balancing for writes.
dsconfig create-criteria-based-load-balancing-algorithm \
     --algorithm-name failover-for-writes \
     --set load-balancing-algorithm:dc_example_dc_com-failover \
     --set "request-criteria:Write Requests"

# Create a proxying request processor instance that uses criteria-based load
# balancing to ensure that writes use a failover strategy, and the remaining
# read operations use a default fewest operations strategy.
dsconfig create-request-processor \
     --processor-name dc_example_dc_com \
     --type proxying \
     --set load-balancing-algorithm:dc_example_dc_com-fewest-operations \
     --set criteria-based-load-balancing-algorithm:failover-for-writes

# Create a subtree view to ensure that operations below "dc=example,dc=com"
# will use the above proxying request processor.
dsconfig create-subtree-view \
     --view-name dc_example_dc_com \
     --set "base-dn:dc=example,dc=com" \
     --set request-processor:dc_example_dc_com

# Update the default client connection policy to include the above subtree
# view.
dsconfig set-client-connection-policy-prop \
     --policy-name default \
     --add subtree-view:dc_example_dc_com
```

Associating PingDirectory Server instances with the appropriate load-balancing algorithms

When configured with an LDAP external server template, either the fewest operations algorithm or the failover load-balancing algorithm will use the topology registry to determine which Directory Server instances should be accessible through the server. To accomplish this, the load-balancing algorithm searches for directory-server-instance objects with a load-balancing-algorithm-name property that matches the name of the load-balancing algorithm. For example, in the following sample configuration, the load-balancing algorithms are named dc_example_dc_com-fewest-operations and dc_example_dc_com-failover, so if you want a Directory Server instance to be accessible through those load-balancing algorithms, add those values to the load-balancing-algorithm-name property in the topology registry's entry for that instance. You can do this with a configuration change similar to the following:

```
dsconfig set-server-instance-prop \
```

```
    --instance-name ds1.example.com:636 \
    --add load-balancing-algorithm-name:dc_example_dc_com-fewest-operations
\
    --add load-balancing-algorithm-name:dc_example_dc_com-failover
```

The PingDirectoryProxy Server automatically detects and reacts to applicable changes in the topology registry. This includes the following::

- Adding a new Directory Server instance to the topology. If this instance has any load-balancing-algorithm-name values that match the names of any load-balancing algorithms that are configured for automatic backend server discovery, the new instance is immediately eligible to process client requests (as soon as the instanced satisfies all of the configured health checks).
- Removing an existing Directory Server instance. If this instance had been associated with any load-balancing algorithms, then it is no longer eligible to receive requests through these load-balancing algorithms.
- Modifying an existing Directory Server instance to add one or more values to the load-balancing-algorithm-name property. If any of the new values matches the name of any of the configured load-balancing algorithms, the instance is immediately eligible to process client requests.
- Modifying an existing Directory Server instance to remove one or more values from the load-balancing-algorithm-name property. If any of the removed values matches the name of any of the configured load-balancing algorithms, then the instance is no longer eligible to receive requests through these load-balancing algorithms.

Automatic backend server discovery with entry balancing

Entry balancing allows directory data to be split across multiple mutually exclusive sets of backend servers. This can provide better scalability by allowing a data set to be fully cached even if it is too large to fit in the memory of any single system. It can also provide better write performance by spreading entries into multiple independent replication sets.

Entry balancing breaks the data into backend sets. Each backend set has its own proxying request processor, which has its own load-balancing configuration (which may optionally include criteria-based load balancing). Each of these load-balancing algorithms may be configured to use automatic backend set discovery in exactly the same way as in a non-entry-balanced configuration. The backend sets must be explicitly configured, but the servers within each set may be automatically determined from the information in the topology registry.

Creating a standard multi-location deployment

In this example deployment, PingDirectoryProxy Server will be deployed in the data centers of two geographic locations: east and west. All LDAP external servers in this deployment are PingDirectory Servers. The directory servers in the eastern city are assigned to the location named east, and the directory servers in the western city are assigned to the location named west.

ⓘ **Note:** Password policies should be kept synchronized across all PingDirectory Server and Directory Proxy Server instances. See the *PingDirectory Server Administration Guide* for details about configuring password policies.

This example refers to four PingDirectory Server instances in two locations with replication of the `dc=example,dc=com` base DN enabled:

- ds-east-01.example.com
- ds-east-02.example.com
- ds-west-01.example.com
- ds-west-01.example.com

We will configure four Directory Proxy Server instances:

- proxy-east-01.example.com

- proxy-east-02.example.com
- proxy-west-01.example.com
- proxy-west-02.example.com

Overview of the deployment steps

In this deployment scenario, we will take the following steps:

- Install the first Directory Proxy Server in east location using the `setup` or `setup.bat` file included in the zip installation file.
- Use the **`create-initial-proxy-config`** tool to provide a proxy user bind DN and password, define locations for each of our data centers, and configure the LDAP external servers in these data centers.
- Test external server communications after initial setup is complete and test a simulated external server failure.
- Install the second proxy server in the east location using the `setup` or `setup.bat` file included in the zip installation file and copy the configuration of the first Directory Proxy Server using the configuration cloning feature.
- Install two Directory Proxy Server instances in the west location, which includes using the setup file and manually setting the location to west using the **`dsconfig`** command, as well as copying the configuration of the Directory Proxy Server using the configuration cloning feature.

After the proxy server has been configured and tested, we then provide a tour of the configuration of each of the proxy server components. These properties can be modified later as needed using the **`dsconfig`** tool.

Installing the first Directory Proxy Server

About this task

To begin with, we have the PingDirectoryProxy Server installation zip file. In this example, we plan to use SSL security, so we also have a keystore certificate database and a pin file that contains the private key password for the keystore. The keystore files are only necessary when using SSL or StartTLS.

In this deployment scenario, the keystore database is assumed to be a Java Keystore (JKS), which can be created by the keytool program. For more information about using the keytool, see the "Security Chapter" in the *PingDirectory Server Administration Guide*.

The PingDirectoryProxy directory contains the following:

```
root@proxy-east-01: ls
ExampleKeystore.jks   ExampleTruststore.jks ExampleKeystore.pin
PingDirectoryProxy-8.0.0.0-with-je.zip
```

The `ExampleKeystore.jks` keystore file contains the private key entry for the `proxy-east-01.example.com` server certificate with the alias `server-cert`. The server certificate, CA, and intermediate signing certificates are all contained in the `ExampleTruststore.jks` file. The password for `ExampleKeystore.jks` is defined in clear text in the corresponding pin file, though the name of the file need not match as it does in our example. The private key password in our example is the same as the password defined for the `ExampleKeystore.jks` keystore.

Steps

1. Unzip the compressed archive file into the PingDirectoryProxy directory and move to this directory.

   ```
   root@proxy-east-01: unzip -q PingDirectoryProxy-<version>-with-je.zip
   root@proxy-east-01: cd PingDirectoryProxy
   ```

2. Because we are configuring SSL security, copy the keystore and pin files into the `config` directory.

   ```
   root@proxy-east01: cp ../*Keystore* config/
   ```

```
root@proxy-east01: cp ../*Truststore* config/
```

3. Next, we install the first proxy server by running the **setup** tool on `proxy-east-01.example.com` as follows:

```
root@proxy-east01: ./setup --no-prompt --acceptLicense \
--ldapPort 389 --rootUserPassword pass \
--aggressiveJVMTuning --maxHeapSize 1g \
--enableStartTLS --ldapsPort 636 \
--useJavaKeystore config/ExampleKeystore.jks \
--keyStorePasswordFile config/ExampleKeystore.pin \
--certNickname server-cert \
--useJavaTrustStore config/ExampleTruststore.jks
```

   New keystore password files are created in `config/keystore.pin`. The original file, `config/ExampleKeystore.pin`, is no longer needed.

4. If you are not using SSL or StartTLS, then the SSL arguments are not necessary as follows:

```
root@proxy-east01: ./setup --no-prompt --acceptLicense \
--ldapPort 389 --rootUserPassword pass \
--aggressiveJVMTuning --maxHeapSize 1g
```

   Once you have installed the Directory Proxy Server, you can configure it using the **create-initial-proxy-config** tool as presented in the next section.

Configuring the first Directory Proxy Server

About this task

Once the Directory Proxy Server has been installed, it can be automatically configured using the **create-initial-proxy-config** tool. This tool can only be used once for this initial configuration, after which we will have to use **dsconfig** to make any changes to our proxy server configuration.

Configuring the Directory Proxy Server with the **create-initial-proxy-config** tool involves the following steps:

- Providing a Directory Proxy Server base DN and password.
- Defining locations for each of our data centers, east and west.
- Configuring the LDAP external server in the east location.
- Configuring the LDAP external servers in the west location.
- Applying the changes to the Directory Proxy Server.

Steps

1. Once we have completed setup, we run the **create-initial-proxy-config** tool as follows:

```
root@proxy-east01: bin/create-initial-proxy-config
```

2. Provide the bind DN and password that the Directory Proxy Server will use to authenticate to the backend PingDirectory Server instances. The **create-initial-proxy-config** tool requires that the same bind DN and password be used to authenticate to all of the backend servers. All Directory Proxy Server instances have identical proxy user accounts and passwords. If necessary, the proxy user account password can be defined differently for each external server using **dsconfig** after the **create-initial-proxy-config** tool has been executed.
3. Specify the type of external server communication security that will be used to communicate with the PingDirectory Server instances. For this example, enter the option for 'None'.
4. Specify the base DNs of the PingDirectory Server instances that the Directory Proxy Server will access. For this example, use `dc=example,dc=com`.
5. Enter any other base DNs of the PingDirectory Server instances that will be accessed through the proxy server. Because we are only using one proxy base DN, press **Enter** to finished.

Defining locations

About this task

Next, we define our first location, east, to accommodate the servers in our deployment located on the East Coast of the United States.

To define proxy locations:

Steps

1. Continuing from the same **`create-initial-proxy-config`** session, enter a location name for the Directory Proxy Server. In this example, enter `east`, and then press **Enter**.
2. Define a location named `west` for the servers in our deployment located on the West Coast. Press **Enter** when finished.
3. Select the location that contains the Directory Proxy Server itself. The Directory Proxy Server is located in the east.

Configuring the external servers in the east and west locations

Once the locations have been defined, we need to identify the directory servers. First, we define one of the servers in the east location.

Configuring the external servers in the east location

Steps

1. Define one of the servers in the east location by entering the host name and port of the server. For this example, enter `ds-east-01.example.com:389`.

```
>>>> External Servers

External Servers identify directory server instances including
host, port, and authentication information.

Enter the host and port (host:port) of the first directory server
in 'east'

    b)  back
    q)  quit

Enter a host:port or choose a menu item [localhost:389]: ds-
east-01.example.com:389
```

2. Enter the option to prepare the server and all subsequent servers. Preparing the servers involves testing the connections to these servers and sets up the `cn=Proxy User` account on the Directory Proxy Server.
3. Enter the DN of the account with which to manage the `cn=Proxy User,cn=Root DNs,cn=config` account. For this example, use the default, `cn=Directory Manager`.
4. Repeat the previous steps to prepare the other server in the east location, ds-east-02.example.com.
5. Press **Enter** to complete preparing the servers.

Configuring the external servers in the west location

About this task

The same process used for the east location is used to define the LDAP external servers for the west location.

Steps

1. Define the first external server, ds-west-01.example.com.
2. Define the second server in the west location, ds-west-02.example.com.
3. Press **Enter** when finished.

Apply the configuration to the Directory Proxy Server

About this task

Next, we review the configuration summary. Once we have confirmed that the changes are correct, we press **Enter** to write the configuration.

To apply the changes to the Directory Proxy Server:

Steps

1. During the configuration process, the **`create-initial-proxy-config`** tool writes the configuration settings to a **`dsconfig`** batch file, which will then be applied to the Directory Proxy Server. The batch file can be reused to configure other servers. On the final step, the **`create-initial-proxy-config`** tool presents a configuration summary. Review the configuration and then apply the changes to the Directory Proxy Server. Press **Enter** to write the configuration to the server.
2. On the final confirmation prompt, press **Enter** to apply the changes to the proxy server, and then enter the LDAP connection parameters to the server. Once the changes have been applied, the **`create-initial-proxy-config`** tool cannot be used to configure this proxy server again.

Configuring additional Directory Proxy Server instances

About this task

We install and configure the second Directory Proxy Server by running the **`setup`** tool on proxy-east-02.example.com.

Steps

1. Copy the keystore and pin files into the `config` directory for the proxy-east-02.example.com server.

```
root@proxy-east-02: cp ../*Keystore* config/
root@proxy-east-02: cp ../*Truststore* config/
```

2. Install the second Directory Proxy Server by running the **`setup`** tool on proxy-east-02.example.com as follows:

```
root@proxy-east-02: ./setup --no-prompt \
--listenAddress proxy-east-02.example.com \
--ldapPort 389 --enableStartTLS --ldapsPort 636 \
--useJavaKeystore config/ExampleKeystore.jks \
--keyStorePasswordFile config/ExampleKeystore.pin \
--certNickName server-cert \
--useJavaTrustStore config/ExampleTruststore.jks \
--rootUserPassword pass --acceptLicense \
--aggressiveJVMTuning --maxHeapSize 1g \
--localHostName proxy-east-02.example.com \
--peerHostName proxy-east-01.example.com \
--peerPort 389 --location east
```

3. Configure the third Directory Proxy Server, proxy-west-01.example.com in the same way as shown in the previous step. First, copy the keystore and pin files into the `config` directory.

```
root@proxy-west-01: cp ../*Keystore* config/
root@proxy-west-01: cp ../*Truststore* config/
```

4. Run the **setup** tool on proxy-west-01.example.com as follows:

```
root@proxy-west-01: ./setup --no-prompt \
--listenAddress proxy-west-01.example.com \
--ldapPort 389 --enableStartTLS --ldapsPort 636 \
--useJavaKeystore config/ExampleKeystore.jks \
--keyStorePasswordFile config/ExampleKeystore.pin \
--certNickName server-cert \
--useJavaTrustStore config/ExampleTruststore.jks \
--rootUserPassword pass --acceptLicense \
--aggressiveJVMTuning --maxHeapSize 1g \
--localHostName proxy-west-01.example.com \
--peerHostName proxy-east-01.example.com \
--peerPort 389 --location west
```

5. Finally, repeat steps 3 and 4 to install the last Directory Proxy Server by first copying the keystore and pin files to the `config` directory and then running the **setup** command.

   At this point, all proxies have the same Admin Data backend and have the `all-servers` group defined as their configuration-server-group in the Directory Proxy Server Global Configuration object. When making a change to a Directory Proxy Server using the **dsconfig** command-line tool or the Administrative Console, you will have the choice to apply the changes locally only or to all proxies in the `all-servers` group.

Testing external server communications after initial setup

About this task

After setting up the basic deployment scenario, the communication between the proxies and the LDAP external servers can be tested using a feature in the proxy server in combination with an LDAP search.

After initial setup, the Directory Proxy Server exposes a special search base DN for testing external server connectivity, called the `backend server pass-through` subtree view. While disabled by default, you can enable this feature using **dsconfig** in the Client Connection Policy menu. Set the value of the `backend-server-passthrough-subtree-views` property to `TRUE`.

Steps

1. Run **dsconfig** to set the `include-backend-server-passthrough-subtree-views` property to `TRUE`.

```
root@proxy-east-01: dsconfig set-client-connection-policy-prop \
--policy-name default \
--set include-backend-server-passthrough-subtree-views:true
```

   Once set to true, an LDAP search against the Directory Proxy Server with the base DN `dc=example,dc=com,ds-backend-server=ds-east-02.example.com:389` instructs the Directory Proxy Server to perform the search against the `ds-east-02.example.com:389` external server with the base DN set to `dc=example,dc=com`. The value of `ds-backend-server` should be the name of the configuration object representing the external server. Depending on your naming scheme, this name may not be a `host:port` combination.

2. Run **ldapsearch** to fetch the `dc=example,dc=com` entry from the `ds-east-01.example.com` server. Perform this search on each external server to determine if external server communication has been configured correctly on the Directory Proxy Server.

```
root@proxy-east-01: bin/ldapsearch \
--bindDN "cn=Directory Manager" \
--bindPassword password \
--baseDN "dc=example,dc=com,ds-backend-server=ds-east-01.example.com:389" \
--searchScope base --useStartTLS "(objectclass=*)"
```

3. You can also use this special subtree view to track the operations performed on each external server to help determine load balancing requirements. This LDAP search can be run with the base DN values for the `ds-east-01` and `ds-east-02` servers to track the distribution of search and bind requests over time. These statistics are reset to zero when the server restarts. The following example searches an external server's monitor entry to display operation statistics:

```
root@proxy-east-01: bin/ldapsearch \
--bindDN "cn=directory manager" \
--bindPassword password \
--baseDN "cn=monitor,ds-backend-server=ds-east-02.example.com:389" \
--searchScope sub --useStartTLS "(cn=ldap*statistics)"

dn: cn=LDAP Connection Handler 192.168.1.203 port 389
Statistics,cn=monitor,ds-backend-server=ds-east-02.example.com:389

objectClass: top
objectClass: ds-monitor-entry
objectClass: ds-ldap-statistics-monitor-entry
objectClass: extensibleObject
cn: LDAP Connection Handler 192.168.1.203 port 389
Statistics
connectionsEstablished: 3004
connectionsClosed: 2990
bytesRead: 658483
bytesWritten: 2061549
ldapMessagesRead: 17278
ldapMessagesWritten: 22611
operationsAbandoned: 0
operationsInitiated: 17278
operationsCompleted: 14241
abandonRequests: 22
addRequests: 1
addResponses: 1
bindRequests: 3006
bindResponses: 3006
compareRequests: 0
compareResponses: 0
deleteRequests: 0
deleteResponses: 0
extendedRequests: 2987
extendedResponses: 2987
modifyRequests: 1
modifyResponses: 1
modifyDNRequests: 0
modifyDNResponses: 0
searchRequests: 8271
searchResultEntries: 8370
searchResultReferences: 0
searchResultsDone: 8246
unbindRequests: 2990
```

Testing a simulated external server failure

About this task

Once you have tested connectivity, run a simulated failure of a load-balanced external server to verify that the Directory Proxy Server redirects LDAP requests appropriately. In this procedure, we stop the ds-east-01.example.com:389 server instance and test searches through proxy-east-01.example.com.

Steps

1. First, perform several searches against the Directory Proxy Server. Verify activity in each of the servers in the east location, ds-east-01 and ds-east-02, by looking at the access logs. Because we used the default load balancing algorithm of fewest operations, it is likely that all of the searches will go to only one of the proxies. The following simple search can be repeated as needed:

```
root@proxy-east-01: bin/ldapsearch \
--bindDN "cn=Directory Manager" \
--bindPassword password --baseDN "dc=example,dc=com" \
--searchScope base --useStartTLS "(objectclass=*)"
```

2. Next, stop the Directory Server instance on ds-east-01.example.com using the **`stop-server`** command and immediately retry the above searches. There should be no errors or noticeable delay in processing the search.

```
root@ds-east-01: bin/stop-server

root@proxy-east-01: bin/ldapsearch \
--bindDN "cn=Directory Manager" \
--bindPassword password --baseDN "dc=example,dc=com" \
--searchScope base --useStartTLS "(objectclass=*)"
```

3. Restart the Directory Proxy Server instance on ds-east-01.example.com. Check the access log to confirm that the Directory Proxy Server started to include the ds-east-01 server in load-balancing within 30 seconds. The default time is 30 seconds, though you can change this default if desired.

Expanding the deployment

In the following example deployment, the PingDirectory Server is deployed in a third, centrally-located data center. The directory servers in the central city is assigned to a new location named central. The proxies will use StartTLS to communicate with the directory servers in the central region.

ⓘ **Note:** Other than the ability to add to the Directory Proxy Server's truststore, the **`prepare-external-server`** tool does not alter the Directory Proxy Server configuration in any way.

The Directory Proxy Server itself, installed on proxy-east-01.example.com, remains in the East location. This example will reconfigure load balancing between the six directory servers in three locations:

- ds-east-01.example.com
- ds-east-02.example.com
- ds-west-01.example.com
- ds-west-02.example.com
- ds-central-01.example.com
- ds-central-02.example.com

Overview of deployment steps

In this deployment scenario, we will take the following steps:

- Prepare the new external servers using the **`prepare-external-server`** tool.
- Use the **`dsconfig`** tool to configure the new LDAP external servers in the central data center and reconfigure the load-balancing algorithm to take these servers into account.
- Test external server communications after the servers have been configured and test a simulated external server failure.

Preparing two new external servers using the prepare-external-server tool

About this task

First, we prepare the external directory servers, ds-central-01 and ds-central-02, by creating the proxy user account and the supporting access rules. In this example, we will connect to the ds-central-01 PingDirectory Server using StartTLS. Because we are using StartTLS, we need to capture the ds-central-01 server's certificate and put it in the trust store on our Directory Proxy Server instance.

The **`prepare-external-server`** tool is located in the `bin` or `bat` directory of the server root directory, PingDirectoryProxy. In this example, we run the tool on the ds-east-01 instance of the Directory Proxy Server.

Steps

1. Run the **`prepare-external-server`** tool to prepare the two new servers. On the first attempted bind to the server, the tool will report a "failed to bind" message as it cannot bind to the `cn=Proxy User` entry due to its not being created yet. The tool sets up the `cn=Proxy User` entry so that the Directory Proxy Server can access it and tests the communication settings to the server.

```
root@proxy-east-01: ./prepare-external-server \
--hostname ds-central-01.example.com --port 389 \
--baseDN dc=example,dc=com \
--proxyBindPassword password  \
--useStartTLS \
--proxyTrustStorePath ../config/ExampleTruststore.jks

Failed to bind as 'cn=Proxy User'

Would you like to create or modify root user 'cn=Proxy User" so that it is
available for this Directory Proxy Server? (yes / no)[yes]:

Enter the DN of an account on ds-central-01:389 with which to create or
 manage the 'cn=Proxy User'
account [cn=Directory Manager]:

Enter the password for 'cn=Directory Manager':

Created 'cn=Proxy User,cn=Root DNs,cn=config'
Testing 'cn=Proxy User' privileges ....Done
```

2. Repeat the process on the other new server in the central location, ds-central-02.

> ⓘ **Note:** For entry-balancing deployments, the global base DN is required when using **`prepare-external-server`**.

Adding the new PingDirectory Servers to the Directory Proxy Server

About this task

After preparing the external PingDirectory Servers to communicate with the Directory Proxy Server, we can now add the two servers in the central location to the proxy server instance. Because we have run the **`prepare-external-server`** tool, the two servers have the `cn=Proxy User` entry configured.

Steps

- Run the **`dsconfig`** tool, which is located in the `bin` or `bat` directory of the server root directory, PingDirectoryProxy.

```
root@proxy-east-01:./dsconfig
```

```
>>>> Specify LDAP connection parameters

Directory Proxy Server host name or IP address [localhost]:

How do you want to connect to the Directory Proxy Server at
localhost?

    1)  LDAP
    2)  LDAP with SSL
    3)  LDAP with StartTLS

Enter choice [1]: 1

Directory Proxy Server at localhost port number [389]:
Administrator user bind DN [cn=Directory Manager]:
Password for user 'cn=Directory Manager':
```

Adding new locations

About this task

First, we add a new central location, to which our new PingDirectory Servers will be added.

The following steps show how to add the new servers to a new location using **dsconfig** interactive.

Steps

1. Run **dsconfig** and enter the LDAP connection parameters when prompted.

   ```
   $ bin/dsconfig
   ```
2. On the main menu, enter the number corresponding to **Location**.
3. On the Location menu, enter the number corresponding to creating a new location.
4. Enter the option to create a new location from scratch.
5. Configure the `preferred-failover-location` property of the new location so that this location fails over first to the east location and then to the west location, should all of the servers in the central location become unavailable.
6. Add the east and west locations as values of the property, specifying them in the order that they will be used for failover.
7. Confirm that these are the correct values and finish configuring the location.

Editing the existing locations

About this task

Next, we edit the existing east and west locations to include the new central location in their failover logic. The new failover logic will be based on geographic distance, so that the east location will first fail over to central and then the west location.

The following example procedure uses **dsconfig** interactive mode to edit the east location.

Steps

1. Run **dsconfig** and enter the LDAP connection parameters when prompted.
2. On the Directory Proxy Server console configuration menu, enter the number corresponding to Location.
3. On the Location menu, enter the number corresponding to viewing and editing an existing location. Then, enter the number corresponding to the Location to be changed.

4. Remove the west location from the `preferred-failover-location` property. It will be added later.
5. Add a new value to the `preferred-failover-location` property.
6. Select the values of the new failover locations for the east.
7. Confirm the new configuration information and save the changes.
8. Repeat steps 2-7 to reconfigure the failover logic for the west location to include the new central location.
9. List the locations to confirm that the new location was added correctly.

Adding new health checks for the central servers

About this task

Next, we must add new health checks for the two new servers.

Steps

1. Run **`dsconfig`** and enter the LDAP connection parameters when prompted.
2. Select the number corresponding to creating a new health check.
3. Enter the option to use an existing health check as a template.
4. Enter the number corresponding to the ds-east-01 health check to use it as a template for the new health check.
5. Name the new health check using the same naming strategy established for the other servers in the deployment. As this health check is for the ds-central-01 server, the name takes the following format:

```
>>>> Enter a name for the Search LDAP Health Check that you want to create:
ds-central-01.example.com:389_dc_example_dc_com-search-health-check
```

6. Review the configuration properties and then enter `f` to finish configuring the new health check and save changes.
7. Repeat steps 2-6 to create another new health check for the `ds-central-02` server.

Adding new external servers

About this task

Add new external servers by selecting “External Server” from the main menu.

Steps

1. Run **`dsconfig`** and enter the LDAP connection parameters when prompted.
2. On the External Server menu, enter the number corresponding to "Create a new External Server".
3. Base the configuration of the new external server on the existing configuration of the ds-east-01 server. Enter `t` to use an existing External Server as a template.
4. Enter the number to base the configuration of the new server on the configuration of the ds-east-01 server.
5. Enter a name for the new ds-central-01 server that complies with the naming strategy.

```
>>>> Enter a name for the Ping Identity DS External Server that you
want to create: ds-central-01.example.com:389
```

6. Enter the value of the `server-host-name` property.
7. Review and modify the configuration properties of the external server.
8. On the External Server menu, change the `server-host-name` property to reflect the name of the `ds-central-01` server.
9. On the External Server menu, change the `location` property to reflect the central location.

10. Change the `health-check` property to reflect the new health check created for the ds-central-01 server in the previous section.
11. On the 'health-check' Property menu, enter the number to remove one or more values.
12. Add the health-check created in the previous section.
13. Select the health check associated with the ds-central-01 server.
14. Press **Enter** to use the value associated with ds-central-01 health check.
15. Review the configuration of the new external server and enter `f` to create the server.
16. Repeat these steps to add the new `ds-central-02` external server.

Modifying the load-balancing algorithm

About this task

To modify the existing load-balancing algorithm to include the newly created servers, select “Load-Balancing Algorithm” from the main menu.

Steps

1. Run **`dsconfig`** and enter the LDAP connection parameters when prompted.
2. Choose the option for Load-Balancing Algorithm.
3. On the Load-Balancing Algorithm menu, enter the number corresponding to "View and edit an existing Load-Balancing Algorithm".
4. Add the ds-central-01 and ds-central-02 servers to the `backend-server` configuration property.
5. On the backend-server property menu, enter the number corresponding to adding one or more values.
6. Select the external servers to add. In this example, select ds-central-01.example.com and ds-central-02.example.com.
7. Review the changes made to the load-balancing algorithm’s configuration properties, and enter `f` to save changes.

   The change has been saved and applied to the Directory Proxy Server. The load-balancing algorithm is referenced in the load-balancing-algorithm property of the request processor used by this Directory Proxy Server.
8. To view this property, go to the main menu and select the Request Processor option.
9. On the Request Processor menu, enter the number corresponding to view and edit an existing request processor.
10. Select the request process used by the Directory Proxy Server, and review the configuration properties.

    This request processor is used by the subtree view serviced by the Directory Proxy Server, which is in turn referenced by the client connection policy.

> ⓘ **Note:** The changes made in this procedure are already in effect. The Directory Proxy Server does not have to be restarted.

Testing external server communication

About this task

After adding and configuring the new external servers, test the communication between the Directory Proxy Server and the LDAP external servers using the `include-backend-server-passthrough-subtree-views` property of the Directory Proxy Server in combination with an LDAP search. For more information about this option, see *Testing External Server Communications* on page 190.

Steps

- Run the **`ldapseasrch`** command to test communications on the ds-central-01 serverTask.

```
root@proxy-east-01: bin/ldapsearch --port 389 --bindDN "cn=directory
 manager" \
--bindPassword password \
--baseDN "dc=example,dc=com,ds-backend-server=ds-central-01.example.com:389"
 \
--searchScope base "(objectclass=*)"
```

  You can repeat this search on the ds-central-02 server, to confirm that the server returns the entry as expected.

Testing a simulated external server failure

About this task

Once you have tested connectivity, run a simulated failure of a load-balanced external server to verify that the Directory Proxy Server redirects LDAP requests appropriately. We stop the ds-east-01.example.com:389 and ds-east-02.example.com:389 server instances and test searches through proxy-east-01.example.com.

Steps

1. We stop the ds-east-01.example.com:389 and ds-east-02.example.com:389 server instances and test searches through proxy-east-01.example.com.
2. Perform several searches against the Directory Proxy Server. Verify activity in each of the servers in the east location, ds-east-01 and ds-east-02, by looking at the access logs. The following simple search can be repeated as needed:

```
root@proxy-east-01: bin/ldapsearch --bindDN "cn=Directory Manager" \
--bindPassword password --baseDN "dc=example,dc=com" \
--searchScope base --useStartTLS "(objectclass=*)"
```

3. Next, stop the Directory Server instance on ds-east-01.example.com and ds-east-02.example.com using the stop-server command and immediately retry the above searches. There should be no errors or noticeable delay in processing the search.

```
root@proxy-east-01: bin/stop-server

root@proxy-east-01: bin/ldapsearch \
--bindDN "cn=Directory Manager" --bindPassword password \
--baseDN "dc=example,dc=com" --searchScope base --useStartTLS \
"(objectclass=*)"
```

4. Check the access log to confirm that requests made to these servers are routed to the central servers, as these servers are the first failover location in the failover list for the ds-east-01 and ds-east-02 servers.
5. Restart the Directory Server instance on ds-east-01.example.com and ds-east-02.example.com. Check their access logs to ensure that traffic is redirected back from the failover servers.

Merging two data sets using proxy transformations

In the following example, the Example.com company acquires Sample Corporation. During the merger, Example.com migrates data from Sample's `o=sample` rooted directory, converting Sample's sampleAccount auxiliary object class usage to Example.com's exampleAccount object class for entries rooted under `dc=example,dc=com`. Knowing that it can take considerable time for Sample's directory clients to become aware of the new DIT and schema, proxy data transformations are created to give the Sample clients as consistent a view of the data as possible during the migratory period. These

transformations allow the clients to search and modify entries under `o=sample` using the Sample Corp. schema.

Overview of the attribute and DN mapping

To achieve the merger of the two data sets, we create proxy transformations that map the Sample source attributes to Example.com target attributes as described in Table 9-1, “Attribute Mapping”. The Example.com schema already defines an attribute to contain the RDN of user entries, called `uid`. However, Example.com chooses to create two new attributes within its `exampleAccount` object class to accommodate two attributes in the Sample schema for representing the region and the DN of linked accounts.

During the merger, Example.com decides to re-parent Sample’s customer entries, which are defined under two different subtrees, ou=east,o=sample and ou=west,o=sample, placing them under Example.com’s `ou=people,dc=example,dc=com` subtree. Associated proxy transformations are described in Table 9-2, "DN Mappings". In this process, Example.com collapses the Sample tree, moving entries from the east and west region under a single DN, `dc=example,dc=com`. The DN proxy transformations assume that all Sample users have been co-located under this single Example.com subtree.

Attribute Mapping

Sample Attribute	Example.com Attribute	Description
sampleID	uid	RDN of user entries
sampleRegion	exSampleRegion	String value representing the region
sampleLinkedAccounts	exSampleLinkedAccounts	DN value

Legacy Sample LDAP applications searching for entries in either the Sample base DN `ou=east,o=sample` or `ou=west,o=sample` will be successfully serviced, though there will be one or more differences in the user entries seen by the Sample legacy applications. Since the Example.com Directory Server has no knowledge of the Sample user’s former `ou=east` or `ou=west` association, search results for client searching under `o=sample` will return a DN that may differ from the original search base. For instance, a search for `sampleID=abc123` under `ou=west,o=sample` may return the user entry for abc123 with the DN of `sampleID=abc123,ou=east,o=sample`. The following table illustrates the mapping DNs.

DN Mapping

Sample DN	Example.com DN
ou=east,o=sample	dc=example,dc=com
ou=west,o=sample	dc=example,dc=com
o=sample	dc=example,dc=com

About mapping multiple source DNs to the same target DN

Some complications exist when defining multiple DN mappings that are used for the same request processor and the same source or target DN (or that have source or target DNs that are hierarchically related). The client request may not include enough information to disambiguate and determine the proper rule to follow.

Several solutions exist to avoid problems of disambiguation. If the client does not need to be able to see all mappings at the same time, then a new client connection policy can be created to use connection criteria that select the set of mappings applied to the client based on information such as the IP address or bind DN. Each client connection policy would have separated subtree views with separate proxying request processors that reference the appropriate transformation for that client.

Alternatively, if it is unnecessary to search under the o=sample base DN, then separate subtree views can be created in the same client connection policy. For example, one subtree view would be created for `ou=east,o=sample` and one for `ou=west,o=sample`. Each subtree view is then associated with its own proxying request processor, one for ou=east requests and one for ou=west requests.

Example of a migrated sample customer entry

The following example is an example of a Sample customer entry that has been migrated to the Example.com database. The user entry is defined in the Example.com Directory Server's database as follows. The attributes that have undergone a proxy transformation are marked in bold. Note that this view is how the entry appears to search requests under the `dc=example,dc=com` base DN.

```
dn: uid=scase,ou=People,dc=example,dc=com
objectClass: person
objectClass: inetOrgPerson
objectClass: organizationalPerson
objectClass: exampleAccount
objectClass: top
description: A customer account migrated from Sample merger
uid: scase
exAccountNumber: 234098
exSampleRegion: east
exSampleLinkedAccounts: uid=jcase,ou=people,dc=example,dc=com
userPassword: password
givenName: Sterling
cn: Sterling Case
sn: Case
telephoneNumber: +1 804 094 3356
street: 00468 Second Street
l: Arlington
mail: sterlingcase@maildomain.com st: VA
```

The following examples shows what the Directory Proxy Server returns to LDAP clients who have requested the entry when searching under the o=sample base DN. Note that the DN returned includes ou=east, even though this branch does not exist in the Example.com DIT. It also returns the attribute names as they are defined in the Sample schema.

```
dn:  sampleID=scase,ou=east,o=sample
objectClass: person
objectClass: inetOrgPerson
objectClass: organizationalPerson
objectClass: exampleAccount
objectClass: top
description: A customer account migrated from Sample merger
uid: scase
exAccountNumber: 234098
exSampleRegion: east
exSampleLinkedAccounts: sampleID=jcase,ou=people,dc=example,dc=com
userPassword: password
givenName: Sterling
cn: Sterling Case
sn: Case
telephoneNumber: +1 804 094 3356
street: 00468 Second Street
l: Arlington
mail: sterlingcase@maildomain.com st: VA
```

Overview of deployment steps

In this deployment scenario, we will take the following steps:

- Install any necessary schema on the Directory Proxy Server.
- Create three attribute mapping proxy transformations and three DN mapping proxy transformations

- Create a new proxying request processor, using the existing `dc_example_dc_com` request processor as a template.
- Assign the six proxy transformations to the new proxying request processor.
- Create a new subtree view for o=sample that references the new proxying request processor.
- Add the new subtree view to the existing client connection policy.
- Test our configuration by performing some searches on the Sample DIT.

About the schema

The Directory Proxy Server inherits user-defined schema from all external servers by comparing `cn=schema` on these servers at Directory Proxy Server startup and at five minute intervals. As a result, example.com schema does not need to be added manually to the Directory Proxy Server's `config/schema` directory. We assume that the schema for Sample entries has been defined on the external servers with the example.com DIT, requiring no direct schema management on the Directory Proxy Server. The following schema definitions are assumed to exist on the external Directory Server:

```
dn: cn=schema
objectClass: top
objectClass: ldapSubentry
objectClass: subschema
cn: schema
attributeTypes: ( 1.3.6.1.4.1.32473.2.1.1
  NAME 'exAccountNumber'
  SYNTAX 1.3.6.1.4.1.1466.115.121.1.15
  SINGLE-VALUE )
attributeTypes: ( 1.3.6.1.4.1.32473.1.1.3
  NAME 'sampleLinkedAccounts'
  SYNTAX 1.3.6.1.4.1.1466.115.121.1.12 )
attributeTypes: ( 1.3.6.1.4.1.32473.1.1.2
  NAME 'sampleRegion'
  SYNTAX 1.3.6.1.4.1.1466.115.121.1.15
  SINGLE-VALUE )
attributeTypes: ( 1.3.6.1.4.1.32473.1.1.1
  NAME 'sampleID'
  SYNTAX 1.3.6.1.4.1.1466.115.121.1.15
  SINGLE-VALUE )
attributeTypes: ( 1.3.6.1.4.1.32473.2.1.3
  NAME 'exSampleLinkedAccounts'
  SYNTAX 1.3.6.1.4.1.1466.115.121.1.12 )
attributeTypes: ( 1.3.6.1.4.1.32473.2.1.2
  NAME 'exSampleRegion'
  SYNTAX 1.3.6.1.4.1.1466.115.121.1.15
  SINGLE-VALUE )
objectClasses: ( 1.3.6.1.4.1.32473.2.2.1
  NAME 'exampleAccount'
  SUP top
  AUXILIARY
  MAY ( exAccountNumber $
        exSampleRegion $
        exSampleLinkedAccounts $
        sampleID $
        sampleRegion $
        sampleLinkedAccounts ) )
```

The schema file defines some Example.com schema, such as `exAccountNumber` and `exSampleRegion`, and some Sample schema, such as `sampleRegion` and `sampleID`.

Creating proxy transformations

About this task

We create three attribute mapping proxy transformations and three DN mapping proxy transformations. We run the **dsconfig** tool, which is located in the `bin` or `bat` directory of the server root directory, PingDirectoryProxy.

Steps

1. In the main server root directory, PingDirectoryProxy, run the **start-server** command.

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

2. Run **dsconfig** in interactive mode and enter the LDAP connection parameters.
3. On the Configuration main menu, enter the number corresponding to **Proxy Transformation**.

Creating the Attribute Mapping Proxy Transformations

Next, we create the attribute mapping proxy transformations using **dsconfig** interactive. We assume for this example that we are continuing from the previous **dsconfig** session. In the following example, this transformation maps `ou=east,o=sample` in the Sample schema `dc=example,dc=com` in the Example.com schema.

Creating the DN mapping proxy transformations

About this task

Now we create the DN mapping proxy transformations.

Steps

1. On the Proxy Transformation menu, enter the number corresponding to Create a new Proxy Transformation.
2. Enter the option to create a new Proxy Transformation from scratch.
3. Enter the option for "DN Mapping Proxy Transformation."
4. Enter a name for the DN Mapping Proxy Transformation. This transformation maps `ou=east,o=sample` in the Sample schema `dc=example,dc=com` in the Example.com schema.
5. Select `TRUE` to enable the transformation by default.
6. Specify the source DN as it appears in client requests.

```
>>>> Configuring the 'source-dn' property

     Specifies the source DN that may appear in client
     requests which should be remapped to the target DN.
     Note that the source DN must not be equal to the target DN.

     Syntax: DN

Enter a value for the 'source-dn' property:
ou=east,o=sample
```

7. Specify the target DN, where requests for the source DN should be routed.

```
>>>> Configuring the 'target-dn' property

    Specifies the DN to which the source DN should be mapped.
    Note that the target DN must not be equal to the source
    DN.
```

```
    Syntax: DN

Enter a value for the 'target-dn' property: dc=example,dc=com
```

8. Review the configuration properties, and then enter f to create the new DN mapping proxy transformation.
9. using the previous steps, create a new DN mapping proxy transformation that maps `ou=west,o=sample` in the Sample schema to `dc=example,dc=com` in the Example.com schema, and name it `sample_west-to-example`.
10. Finally, create a DN mapping proxy transformation for the base DN of the Sample database.

Creating a request processor to manage the proxy transformations

About this task

Next, we need to create a new proxying request processor that includes our new attribute and DN mapping proxy transformations. We will use the existing `dc_example_dc_com` request processor as a template.

Steps

1. On the Configuration main menu, enter the number corresponding to Request Processor.
2. On the Request Processor menu, enter the number corresponding to "Create a new Request Processor."
3. Choose the option to use the current request processor as a template.
4. Provide a name for the new proxying request processor, such as `o_sample-req-processor`.
5. Review the properties. The load-balancing algorithm is the same as for the previous request processor, though the transformation property must be changed. Enter the number corresponding to the `Transformation` property.
6. Enter the number corresponding to the proxy transformations that we created in the previous sections.
7. Select the attribute mapping proxy transformations first. Next, select the DN mapping proxy transformations. The order of the selection is important because we have related DNs. Begin with the DNs that are lower in the tree first, and finish with the base DN transformation.

```
Select the Proxy Transformations you wish to add:

    1)  sample-to-example       5)  sampleLinkedAccounts-to-
                                    exSampleLinkedAccounts
    2)  sample_east-to-example  6)  sampleRegion-to-
                                    exSampleRegion
    3)  sample_west-to-example  7)  Create a new Proxy
                                    Transformation
    4)  sampleID-to-uid         8)  Add all Proxy Transformations

    ?)  help
    b)  back
    q)  quit

Enter one or more choices separated by commas [b]: 4,5,6,2,3,1
```

8. Confirm that the proxy transformations are listed in the correct order and press **Enter** to accept and use the values.
9. Review the request processor properties, and enter `f` to save changes.

Creating subtree views

About this task

At this stage, we need to configure subtree views for the Directory Proxy Server.

Steps

1. On the Configuration main menu, enter the number corresponding to Subtree View.
2. On the Subtree View menu, enter the number corresponding to "Create a new Subtree View."
3. Enter the option to create the new subtree view from an existing one.
4. Select the `dc_example_dc_com-view` subtree view.
5. Enter a descriptive name for the subtree view configuration.
6. Configure the base DN property of the Sample dataset.
7. Enter the request processor created in the previous section.
8. Review the configuration properties, and enter `f` to save changes.

```
>>>> Configure the properties of the Subtree View

        Property           Value(s)
        -------------------------------------------------
    1)  description        -
    2)  base-dn            "o=sample"
    3)  request-processor  o_sample-req-processor

    ?)  help
    f)  finish - create the new Subtree View
    d)  display the equivalent dsconfig arguments to create this
        object
    b)  back
    q)  quit
```

Editing the client connection policy

About this task

Finally, we edit the client connection policy to add our new `o=sample` subtree view.

Steps

1. On the Configuration main menu, enter the number corresponding to Client Connection Policy.
2. On the Client Connection menu, enter the number corresponding to "Create a new Client Connection."
3. In the configuration properties, select the `subtree-view` property. Enter the number corresponding to "Add one or more values" to add the new subtree view created for the previous example.
4. Select the subtree view that was created in the previous section.

```
Select the Subtree Views you wish to add:

    1)  o_sample-view
    2)  Create a new Subtree View
```

5. Review the subtree views now referenced by the property and press **Enter** to use these values.
6. Review the configuration properties of the client connection policy and enter `f` to save changes.

Testing proxy transformations

About this task

After setting up the deployment scenario, the Directory Proxy Server will now respond to requests to the `dc=example,dc=com` and o=sample base DNs. We now test the service by imitating example client requests to search and modify users.

The following example fetches the user with sampleID=scase under the `ou=east,o=sample` base DN.

Steps

1. Run **`ldapsearch`** to view a Sample entry.

```
root@proxy-east-01: bin/ldapsearch --bindDN "cn=directory manager" \
--bindPassword password --baseDN "ou=east,o=sample" "(sampleID=scase)"
```

```
dn: sampleID=scase,ou=People,ou=east,o=sample
objectClass: person
objectClass: organizationalPerson
objectClass: inetOrgPerson
objectClass: exampleAccount
objectClass: top
description: A customer account migrated from Sample merger
sampleID: scase
userPassword: {SSHA}A5O4RrQHWXc2Ii3btD4exGdP0TVW9VL3CR3ZXA==
exAccountNumber: 234098
givenName: Sterling
cn: Sterling Case
sn: Case
telephoneNumber: +1 804 094 3356
street: 00468 Second Street
mail: sterlingcase@maildomain.com
l: Arlington
st: VA
sampleRegion: east
sampleLinkedAccounts: sampleID=jcase,ou=People,ou=east,o=sample
```

2. Modify the sampleRegion value, changing it to west. To do this, we first create a **`ldapmodify`** input file, called `scase-mod.ldif`, with the following contents:

```
dn: sampleID=scase,ou=People,ou=east,o=sample
changetype: modify
replace: sampleRegion
sampleRegion: west
```

3. Use the file as an argument in the **`ldapmodify`** command as follows.

```
root@proxy-east-01: bin/ldapmodify --bindDN "cn=Directory Manager" \
--bindPassword password --filename scase-mod.ldif
```

```
Processing MODIFY request for sampleID=scase,ou=People, ou=east,o=sample
MODIFY operation successful for DN sampleID=scase,ou=People,
 ou=east,o=sample
```

4. Search for scase's `sampleRegion` value under o=sample, we should see west:

```
root@proxy-east-01: bin/ldapsearch --bindDN "cn=directory manager" \
--bindPassword password --baseDN "o=sample"  "(sampleID=scase)" \
sampleRegion
```

```
dn: sampleID=scase,ou=People,ou=east,o=sample
sampleRegion: west
```

5. Search for scase by `uid` rather than `sampleID`, under the `dc=example,dc=com` base DN. We see the Example.com schema version of the entry:

```
root@proxy-east-01: bin/ldapsearch --bindDN "cn=directory manager" \
--bindPassword password --baseDN "dc=example,dc=com"  "(uid=scase)"
```

```
dn: uid=scase,ou=People,dc=example,dc=com
objectClass: person
objectClass: exampleAccount
objectClass: inetOrgPerson
objectClass: organizationalPerson
```

```
objectClass: top
description: A customer account migrated from Sample merger
uid: scase
userPassword: {SSHA}A5O4RrQHWXc2Ii3btD4exGdP0TVW9VL3CR3ZXA==
exAccountNumber: 234098
givenName: Sterling
cn: Sterling Case
sn: Case
telephoneNumber: +1 804 094 3356
street: 00468 Second Street
mail: sterlingcase@maildomain.com
l: Arlington
st: VA
exSampleRegion: west
exSampleLinkedAccounts: uid=jcase,ou=People,dc=example,dc=com
```

Deploying an Entry-Balancing Directory Proxy Server

You can deploy PingDirectoryProxy Server in a variety of ways, depending upon the needs of your enterprise. This chapter describes and illustrates an entry-balancing deployment scenario.

Deploying an entry-balancing proxy configuration

Entry-balancing is a Directory Proxy Server configuration that allows the entries within a portion of the Directory Information Tree (DIT) to reside on multiple external servers. This configuration is typically useful when the DIT contains many millions of entries, which can be difficult to bring completely into memory for optimal performance. Entry-balancing allows entries under a balancing point base DN to be divided among any number of separate directory servers, making the Directory Proxy Server responsible for intelligently routing requests based on the division.

In this example scenario, the entries in the DIT outside of the balancing point are replicated across all external servers known to the Directory Proxy Server. Replication on the external directory servers must be properly configured before proceeding through this example. The directory servers are expected to contain two replication domains: the global domain, `dc=example,dc=com`, and the balancing point, `ou=people,dc=example,dc=com`.

In this deployment scenario, an `austin-proxy1` instance of the Directory Proxy Server communicates with four external directory servers. The Directory Proxy Server is configured to use entry balancing for the `ou=people,dc=example,dc=com` base DN, with two sets of user entries split beneath it. The first set of user entries is defined in the replicated pair of external servers, `austin-set1.example.com` and `newyork-set1.example.com`. The second set of entries is defined in `austin-set2.example.com` and `newyork-set2.example.com`. The entries in the `dc=example,dc=com` DIT outside of the balancing point base DN are replicated among the four external servers.

The following **`dsreplication status`** output from the PingDirectory Server external servers describes the replication configuration that exists before creating the Directory Proxy Server configuration.

```
--- Replication Status for dc=example,dc=com: Enabled ---

Server : Entries : Backlog : Oldest Backlog Change Age : Generation ID
--------:---------:--------:---------------------------:-------------

austin-set1.example.com:389  : 10003   : 0   : N/A  : 722087263
austin-set2.example.com:389  : 10003   : 0   : N/A  : 722087263
newyork-set1.example.com:389 : 10003   : 0   : N/A  : 722087263
newyork-set2.example.com:389 : 10003   : 0   : N/A  : 722087263

--- Replication Status for ou=people,dc=example,dc=com (Set: dataset1):
 Enabled ---
```

```
Server : Entries : Backlog : Oldest Backlog Change Age : Generation ID
--------:---------:---------:---------------------------:-----------------

austin-set1.example.com:389  : 100001  : 0  : N/A               : 178892712
newyork-set1.example.com:389 : 100001  : 0  : N/A               : 178892712

--- Replication Status for ou=people,dc=example,dc=com (Set: dataset2):
 Enabled ---

Server : Entries : Backlog : Oldest Backlog Change Age : Generation ID
---------:---------:---------:---------------------------:-----------------

austin-set2.example.com:389  : 100001  : 0       : N/A         : 1057593890
newyork-set2.example.com:389 : 100001  : 0       : N/A         : 1057593890
```

Determining how to balance your data

If a single Directory Server instance can hold all of your data, then we recommend storing your data on a single server and replicating for high availability, as this simplifies your deployment. If a single server cannot hold all of your data, then you can spread it across multiple servers in several ways:

- If the data is already broken up by hierarchy and all of the clients understand how to access it that way, the number of top-level branches is small and a single Directory Server instance can hold all of the information within one or more branches. Configure the Directory Proxy Server with multiple base DNs and use simple load-balancing rather than entry balancing to simplify your deployment.
- If simply breaking up the data using the existing hierarchy is not an option, for example if a large number of top-level branches must be configured, then consider using entry balancing. The contents of any single branch still must fit on a given server, because only entries that are immediate subordinates of the entry-balancing base DN may be spread across multiple servers. Any entries that are further subordinates have to be placed in the same directory server instance as their parent.
- If one or more branches are so large that any single Directory Server instance cannot hold all of the data, you need to use entry balancing within that branch to divide the entries among two or more sets of Directory Servers. You may also need to change the way that the data is arranged in the server so that it uses as flat a DIT as possible, which is easier to use in an entry-balancing deployment.

In an entry-balancing deployment, there can be data that is common to all external directory servers outside the balancing point. This data is referred to as the global domain. The Directory Proxy Server entry-balancing configuration will contain at least two subtree views and associated request processors, one for the global domain and one for the entry-balancing domain. In our examples, the global domain is `dc=example,dc=com` and the entry-balancing domain is `ou=people, dc=example,dc=com`. The entry-balancing base DN, `ou=people,dc=example,dc=com`, is also the balancing point.

Entry balancing and ACIs

In an entry-balancing deployment, access control instructions (ACIs) are still configured in the backend Directory Server data. When defining access controls in an entry-balancing deployment, you need to ensure that the data used by the access control rule is available for evaluation on all datasets.

If you use groups for access control and a group contains users from different data sets, then that group must exist on each dataset. For a single ACI to be applicable to entries in all datasets, it must be specified above the entry-balancing point. For example, if an ACI allows access to modify users that are part of group 1, then two things must exist on both data sets:

- Group 1 must exist in the `ou=groups` branch of both datasets.
- The ACI referencing group 1 must exist in the `ou=people` branch or above. The `ou=people` branch entry itself is part of the common data.

The Directory Proxy Server ensures that any changes to entries within the scope of the entry-balancing request processor, but outside the balancing point, are applied to all backend server sets. Any ACI stored at the entry-balancing point will be kept in sync if changes are made through the Directory Proxy Server.

Overview of deployment steps

In this deployment scenario, we will take the following steps:

- Install the Directory Proxy Server on `austin-proxy1`.
- Use the **`create-initial-proxy-config`** tool to provide our initial setup for entry balancing. The initial setup includes defining multiple subtree views and global indexes in support of entry balancing.
- Change the placement algorithm of the `austin-proxy-01` server to use an entry-count placement algorithm. This algorithm is used to select the backend set to which to forward an add request. It looks at the number of entries in the backend sets and forwards the add request to the backend with either the fewest or the most entries, depending on the configuration. You can also configure the placement algorithm to make the decision based on the on-disk database size rather than the number of entries.

Installing the Directory Proxy Server

About this task

We start by configuring the Directory Proxy Server. The four external servers, `austin-set1.example.com`, `newyork-set1.example.com`, `austin-set2.example.com`, and `newyork-set2.example.com`, are running.

Steps

- Run the **`setup`** program in non-interactive mode.

```
root@austin-proxy1: ./setup --acceptLicense \
--listenAddress austin-proxy1.example.com \
--ldapPort 389 --rootUserDN "cn=Directory Manager" \
--rootUserPassword pass --entryBalancing \
--aggressiveJVMTuning --maxHeapSize 2g --no-prompt
```

Configuring the entry-balancing Directory Proxy Server

About this task

Once the Directory Proxy Server has been installed, it can be automatically configured using the **`create-initial-proxy-config`** tool. This tool can only be used once for this initial configuration, after which we will have to use **`dsconfig`** to make any changes to our Directory Proxy Server configuration.

Steps

1. Run the **`create-initial-proxy-config`** tool.

```
root@austin-proxy1: ./bin/create-initial-proxy-config
```

2. Our topology meets the requirements, press **Enter** to continue:

```
Some assumptions are made about the topology to keep
this tool simple:

1) all servers will be accessible via a single user account
2) all servers support the same communication security type
3) all servers are PingDirectoryProxy Servers

If your topology does not have these characteristics you can
use this tool to define a basic configuration and then use the
'dsconfig' tool or the Administrative Console to fine tune the
 configuration.

Would you like to continue? (yes / no) [yes]:
```

3. Provide the external server access credentials. All of our proxies have identical proxy user accounts and passwords.

```
Enter the DN of the proxy user account [cn=Proxy User,cn=Root
 DNs,cn=config]:

Enter the password for 'cn=Proxy User,cn=Root DNs,cn=config':
Confirm the password for 'cn=Proxy User,cn=Root DNs,cn=config':
```

4. Specify the type of security that the Directory Proxy Server will use to communicate with Directory Servers.
5. Enter a base DN of the Directory Server instances that will be accessed by the Directory Proxy Server.
6. Define the balancing point as a separate base DN, which is entry balanced:

```
Enter another base DN of the directory server instances that
will be accessed through the Directory Proxy Server:

    1)Remove dc=example,dc=com

    b)back

    q)quit

Enter a DN or choose a menu item [Press ENTER when finished
entering base DNs]: ou=people,dc=example,dc=com

Are entries within 'ou=people,dc=example,dc=com' split across
multiple servers so that each server stores only a subset of
the entries (i.e. is this base DN 'entry balanced')? (yes / no)
[no]: yes
```

7. In this example, the data in `ou=people,dc=example,dc=com` will be split across two backend sets. Enter 2 to specify that the data will be balanced across two sets of servers.

```
Across how many sets of servers is the data balanced?

    c) cancel creating ou=people,dc=example,dc=com
    q) quit

Enter a number greater than one or choose a menu item: 2
```

8. The balancing point is the same as our base DN, ou=people,dc=example,dc=com., so we use it as the entry balancing base.

```
>>>> Entry Balancing Base

The entry balancing base DN specifies the entry below which the
data is balanced. Entries not below this entry must be duplicated
in all the server sets. If all the entries in the base DN are
distributed the entry balancing base DN is the same as the base DN.

    c) cancel creating ou=people,dc=example,dc=com
    b) back
    q) quit

Enter the entry balancing base DN or choose a menu item
[ou=people,dc=example,dc=com]: ou=people,dc=example,dc=com
```

9. To improve the performance for equality search filters referencing the `uid` attribute, create a `uid` global index. Enter `yes` to add a new attribute to the global index.
10. Specify the `uid` attribute.

```
Enter attributes that you would like to add to the global index:
```

```
    c)cancel creating ou=people,dc=example,dc=com
    b)back
    q)quit

Enter an attribute name or choose a menu item [Press ENTER when
finished entering index attributes]: uid
```

11. To optimize Directory Proxy Server performance from the moment it starts accepting connections, enter the number corresponding to "Yes, and all subsequent attributes."

12. Press **Enter** to finish specifying index attributes.

13. Press **Enter** to enable RDN index priming.

```
Would you like to enable RDN index priming for
'ou=people,dc=example,dc=com'? (yes / no) [yes]:
```

14. Press **Enter** to finish specifying base DNs.

```
Enter another base DN of the directory server instances that
will be accessed through the Directory Proxy Server:

    1) Remove dc=example,dc=com
    2) Remove ou=people,dc=example,dc=com (distributed)

    b)  back
    q)  quit

Enter a DN or choose a menu item [Press ENTER when finished
entering base DNs]:
```

15. The external servers are spread among two locations, New York and Austin. This Directory Proxy Server instance is located in the austin location.

```
A good rule of thumb when naming locations is to use the
name of your data centers or the cities containing them.

    b)  back
    q)  quit

Enter a location name or choose a menu item: austin

    1)  Remove austin

    b)  back
    q)  quit
```

16. Define the newyork location:

```
Enter another location name or choose a menu item [Press ENTER
when finished entering locations]: newyork

    1)  Remove austin
    2)  Remove newyork

    b)  back
    q)  quit

Enter another location name or choose a menu item [Press ENTER
when finished entering locations]:
```

17. Select the austin location for this Directory Proxy Server instance:

```
Choose the location for this Directory Proxy Server

    1) austin
```

```
    2) newyork

    b) back
    q) quit

Enter choice [1]:
```

18. Specify the LDAP external server instances associated with this location.

```
Enter the host and port (host:port) of the first directory server
in 'austin'

     b)  back
     q)  quit

Enter a host:port or choose a menu item [localhost:389]:
austin-set1.example.com:389
```

19. Specify that the `austin-set1` server can handle requests from the global domain and from set 1 restricted domain.

```
Assign server austin-set1.example.com:389 to handle requests for
one or more of the defined sets of data:

    1) dc=example,dc=com
    2) ou=people,dc=example,dc=com; Server Set 1
    3) ou=people,dc=example,dc=com; Server Set 2

Enter one or more choices separated by commas: 1,2
```

20. Enter the number corresponding to "Yes, and all subsequent servers" to prepare the server for access by the Directory Proxy Server.

```
Would you like to prepare austin-set1.example.com:389 for access
by the Directory Proxy Server?

       1)Yes
       2)No
       3)Yes, and all subsequent servers
       4)No, and all subsequent servers

Enter choice [3]:
```

21. Select the entry-balanced data set that the austin-set1 server replicates with other servers.

```
You may choose a single entry-balanced data set with which
austin-set1.example.com:389 will replicate data with other servers

    1) ou=people,dc=example,dc=com; Server Set 1
    2) None, data will not be replicated

Enter choice: 1

Testing connection to austin-set1.example.com:389 ..... Done
Testing 'cn=Proxy User,cn=Root DNs,cn=config' access ....Denied
```

22. Modify the root user for use by the Directory Proxy Server, specifying the directory manager password for the initial creation of the proxy user.

```
Would you like to create or modify root user 'cn=Proxy User,
cn=Root DNs,cn=config' so that it is available for this
Directory Proxy Server? (yes / no) [yes]:

Enter the DN of an account on austin-set1.example.com:389
with which to create or manage the 'cn=Proxy User,cn=Root DNs,
```

```
cn=config' account and configuration [cn=Directory Manager]:

Enter the password for 'cn=Directory Manager':
Created 'cn=Proxy User,cn=Root DNs,cn=config'
Testing 'cn=Proxy User,cn=Root DNs,cn=config'privileges...Done
Setting replication set name .....
```

23. Since the replication set name has already been configured, we do not need to use the name created automatically by the Directory Proxy Server.

```
This server is currently configured for replication set 'dataset1'.
Would you like to reconfigure this server for replication set
'set-1'? (yes / no) [no]:

Setting replication set name ..... Done
Verifying backend 'dc=example,dc=com' ..... Done
Verifying backend 'ou=people,dc=example,dc=com' ..... Done
Testing 'cn=Proxy User' privileges ..... Done
Verifying backend 'dc=example,dc=com' ..... Done
```

24. Define the other Austin and New York servers using the same procedure as in the previous example:

```
Enter another server in 'austin'

    1) Remove austin-set1.example.com:389
    b) back
    q) quit

Enter a host:port or choose a menu item [Press ENTER when
finished entering servers]: austin-set2.example.com:389

Assign server austin-set2.example.com:389 to handle requests
for one or more of the defined sets of data

    1) dc=example,dc=com
    2) ou=people,dc=example,dc=com; Server Set 1
    3) ou=people,dc=example,dc=com; Server Set 2

Enter one or more choices separated by commas: 1,3

You may choose a single entry-balanced data set with which
austin-set2.example.com:389 will replicate data with other
servers

    1) ou=people,dc=example,dc=com; Server Set 2
    2) None, data will not be replicated

Enter choice: 1

Testing connection to austin-set2.example.com:389 ....Done
Testing 'cn=Proxy User,cn=Root DNs,cn=config' access ... Denied

Would you like to create or modify root user 'cn=Proxy User,
cn=Root DNs,cn=config' so that it is available for this
Directory Proxy Server? (yes / no) [yes]:

Would you like to use the previously entered manager credentials
to access all prepared servers? (yes / no) [yes]:

Created 'cn=Proxy User,cn=Root DNs,cn=config'
Testing 'cn=Proxy User,cn=Root DNs,cn=config' privileges...Done
Setting replication set name .....

This server is currently configured for replication set 'dataset2'.
```

```
Would you like to reconfigure this server for replication set 'set-2'?
(yes / no) [no]:

Setting replication set name ..... Done
Verifying backend 'dc=example,dc=com' ..... Done
Verifying backend 'ou=people,dc=example,dc=com' ..... Done

Enter another server in 'austin'

    1) Remove austin-set1.example.com:389
    2) Remove austin-set2.example.com:389

    b) back
    q) quit

Enter a host:port or choose a menu item [Press ENTER when
finished entering servers]:

>>>> >>>> Location 'newyork' Details
 >>>> External Servers

External Servers identify directory server instances including
host, port, and authentication information.

Enter the host and port (host:port) of the first directory server
in 'newyork':

    b) back
    q) quit

Enter a host:port or choose a menu item [localhost:389]:
newyork-set1.example.com:389

Assign server newyork-set1.example.com:389 to handle requests
for one or more of the defined sets of data

    1) dc=example,dc=com
    2) ou=people,dc=example,dc=com; Server Set 1
    3) ou=people,dc=example,dc=com; Server Set 2

Enter one or more choices separated by commas: 1,2

You may choose a single entry-balanced data set with which
newyork-set1.example.com:389 will replicate data with other servers

    1) ou=people,dc=example,dc=com; Server Set 1
    2) None, data will not be replicated

Enter choice: 1

Testing connection to newyork-set1.example.com:389 ....Done
Testing 'cn=Proxy User,cn=Root DNs,cn=config' access ... Denied

Would you like to create or modify root user 'cn=Proxy User,
cn=Root DNs,cn=config' so that it is available for this
Directory Proxy Server? (yes / no) [yes]:

Created 'cn=Proxy User,cn=Root DNs,cn=config'
Testing 'cn=Proxy User,cn=Root DNs,cn=config' privileges...Done
Setting replication set name .....

This server is currently configured for replication set 'dataset1'.
```

```
Would you like to reconfigure this server for replication set
'set-1'? (yes / no) [no]:

Setting replication set name ..... Done
Verifying backend 'dc=example,dc=com' ..... Done
Verifying backend 'ou=people,dc=example,dc=com' ..... Done

Enter another server in 'newyork'

    1) Remove newyork-set1.example.com:389
    b) back
    q) quit

Enter a host:port or choose a menu item [Press ENTER when
finished entering servers]: newyork-set2.example.com:389

Assign server newyork-set2.example.com:389 to handle requests
for one or more of the defined sets of data:

    1) dc=example,dc=com
    2) ou=people,dc=example,dc=com; Server Set 1
    3) ou=people,dc=example,dc=com; Server Set 2

Enter one or more choices separated by commas: 1,3

You may choose a single entry-balanced data set with which
new-york-set2.example.com:389 will replicate data with other servers

    1) ou=people,dc=example,dc=com; Server Set 2
    2) None, data will not be replicated

Enter choice: 1

Testing connection to newyork-set2.example.com:389 ..... Done
Testing 'cn=Proxy User,cn=Root DNs,cn=config' access.... Denied

Would you like to create or modify root user 'cn=Proxy User,
cn=Root DNs,cn=config' so that it is available for this Directory
Proxy Server? (yes / no) [yes]:

Created 'cn=Proxy User,cn=Root DNs,cn=config' Testing
'cn=Proxy User,cn=Root DNs,cn=config' privileges...Done
Setting replication set name .....

This server is currently configured for replication set 'dataset2'.
Would you like to reconfigure this server for replication
set 'set-2'? (yes / no) [no]:

Setting replication set name ..... Done
Verifying backend 'dc=example,dc=com' ..... Done
Verifying backend 'ou=people,dc=example,dc=com' ..... Done

Enter another server in 'newyork'

    1)Remove newyork-set1.example.com:389
    2)Remove newyork-set2.example.com:389

    b)back
    q)quit

Enter a host:port or choose a menu item [Press ENTER when
finished entering servers]:

>>>> >>>> Configuration Summary
```

```
  External Server Security: None
  Proxy User DN: cn=Proxy User,cn=Root DNs,cn=config
  Location austin
    Failover Order: newyork
    Servers: austin-set1.example.com:389,
             austin-set2.example.com:389
  Location newyork
    Failover Order: austin
    Servers: newyork-set1.example.com:389,
             newyork-set2.example.com:389
  Base DN: dc=example,dc=com
    Servers: austin-set1.example.com:389,
             austin-set2.example.com:389,
             newyork-set1.example.com:389,
             newyork-set2.example.com:389
  Base DN:vou=people,dc=example,dc=com
    Entry Balancing Base: ou=people,dc=example,dc=com
    Server Set 1: austin-set1.example.com:389,
                  newyork-set1.example.com:389
    Server Set 2: austin-set2.example.com:389,
                  newyork-set2.example.com:389
    Index Attributes: uid (primed,unique)
    Prime RDN Index: Yes

    NOTE: The Directory Proxy Server must be restarted after
    this tool has completed to have index priming take place

       b) back
       q) quit
       w) write configuration

    Enter choice [w]
    >>>> Write Configuration

    The configuration will be written to a 'dsconfig' batch
    file that can be used to configure other Directory Proxy Servers.

    Writing Directory Proxy Server configuration to /proxy/dps-
cfg.txt.....Done
```

25. Enter `yes` to apply our configuration changes to the Directory Proxy Server.

```
Apply these configuration changes to the local Directory Proxy
Server? (yes /no) [yes]:

How do you want to connect to the Directory Proxy Server on localhost?

    1) LDAP
    2) LDAP with SSL
    3) LDAP with StartTLS

Enter choice [1]:

Administrator user bind DN [cn=Directory Manager]:
Password for user 'cn=Directory Manager':
Creating Locations ..... Done
Updating Failover Locations ..... Done
Updating Global Configuration ..... Done
Creating Health Checks ..... Done
Creating External Servers ..... Done
Creating Load-Balancing Algorithm for dc=example,dc=com .... Done
Creating Request Processor for dc=example,dc=com ..... Done
Creating Subtree View for dc=example,dc=com ..... Done
```

```
Updating Client Connection Policy for dc=example,dc=com ..... Done
Creating Load-Balancing Algorithm for ou=people,dc=example,dc=com; Server
 Set 1 ..... Done
Creating Request Processor for ou=people,dc=example,dc=com; Server Set
 1...Done
Creating Load-Balancing Algorithm for ou=people,dc=example,dc=com; Server
 Set 2 .... Done
Creating Request Processor for ou=people,dc=example,dc=com; Server Set
 2...Done
Creating Entry Balancing Request Processor for
 ou=people,dc=example,dc=com ..... Done
Creating Placement Algorithm for ou=people,dc=example,dc=com .... Done
Creating Global Attribute Indexes for ou=people,dc=example,dc=com ..... Done
Creating Subtree View for ou=people,dc=example,dc=com ..... Done
Updating Client Connection Policy for ou=people,dc=example,dc=com ..... Done

See /logs/create-initial-proxy-config.log for a detailed log of this
 operation

To see basic server configuration status and configuration you can launch /
bin/status
```

Configuring the placement algorithm using a batch file

About this task

Now, we configure the placement algorithm using a batch file. We want to place new entries added through the proxy via LDAP ADD operations into the least used dataset. We do this using an entry-count placement algorithm. To change the placement algorithm from round-robin to entry-count, we first create and enable an entry-count placement algorithm configuration object and then disable the existing round-robin placement algorithm. Our batch file, `dsconfig.post-setup`, contains the **`dsconfig`** commands required to create the entry-count placement algorithm and disable the old round-robin algorithm.

The batch file contains comments to explain each **`dsconfig`** command. Note that in this example, line wrapping is used for clarity. The **`dsconfig`** command requires that the full command be provided on a single line.

The batch file itself looks like the following:

```
root@austin-proxy1:more ../dsconfig.post-setup

# This dsconfig operation creates the entry-count placement
# algorithm with the default behavior of adding entries to the
# smallest backend dataset first.

dsconfig create-placement-algorithm
--processor-name ou_people_dc_example_dc_com-eb-req-processor
--algorithm-name entry-count --type entry-counter --set enabled:true

# Note that once the entry-count placement algorithm is created
# and enabled, we can disable the round-robin algorithm.
# Since an entry-balancing proxy must always have a placement
# algorithm, we add a second algorithm and then disable the
# original round-robin algorithm created during the setup
# procedure.

dsconfig set-placement-algorithm-prop
--processor-name ou_people_dc_example_dc_com-eb-req-processor
--algorithm-name round-robin --set enabled:false

# At this point, LDAP ADD operations will be forwarded to an external
# server representing the dataset with the least number of entries.
```

Steps

- Run the **`dsconfig`** command using the batch file. Once the batch file has executed, a new entry-count placement algorithm, called entry-count, has been created, and the old round-robin placement algorithm, `round-robin`, has been disabled.

```
root@austin-proxy1: bin/dsconfig --no-prompt \
--bindDN "cn=directory manager" --bindPassword password \
--port 389  --batch-file ../dsconfig.post-setup
```

```
Batch file '../dsconfig.post-setup' contains 2 commands

Executing: create-placement-algorithm --no-prompt
--bindDN "cn=directory manager" --bindPassword pass
--port 1389
--processor-name ou_people_dc_example_dc_com-eb-req-processor
--algorithm-name entry-count --type entry-counter --set enabled:true

Executing: delete-placement-algorithm --no-prompt
--bindDN "cn=directory manager" --bindPassword pass
--port 1389
--processor-name ou_people_dc_example_dc_com-eb-req-processor
--algorithm-name round-robin --set enabled:false
```

Rebalancing your entries

If your deployment distributes entries using an entry counter placement algorithm or 3rd party algorithm, you may need to redistribute your entries. For example, imagine that you have an environment that distributes entries across three backends using an entry counter placement algorithm. This algorithm distributes entries to the backend that has the most space. Imagine that the backends all reach their maximum capacity and you decide to add a new backend to the deployment. You need to move the entries from the full backends and distribute them evenly across all the backends, including the new backend.

You might also want to deliberately rebalance your entries to meet the needs of your organization. For example, you can direct entry balancing based on attributes on the entries themselves. You can write a custom algorithm that looks at the value of an attribute that is being modified on the entry. Based on the attribute, you can then put this entry somewhere specific. You might use this feature if you want to have certain entries closer geographically to the client application using them. The geographical information could be included in the entry. Rebalancing would be used to move these entries to the server in the correct geographical location.

You can redistribute entry-balanced entries in two ways:

- **Using dynamic rebalancing.** With dynamic rebalancing, as existing entries get modified, they get moved. You configure dynamic rebalancing in the entry counter placement algorithm.
- **Using the `move-subtree` tool.** This tool can be used to move either small subtrees through a transactional method or to move large subtrees, potentially taking them offline for a short period.

The remainder of this section describes each of these method of entry rebalancing in more detail.

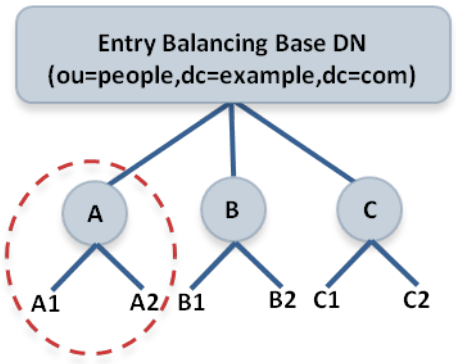
About dynamic rebalancing

During dynamic rebalancing entries get moved as they are modified. You configure dynamic rebalancing on the entry counter placement algorithm or a third-party placement algorithm that supports rebalancing. This algorithm keeps a count of the number of entries or the size of the backend set. You configure dynamic rebalancing using the following parameters:

- **rebalancing-enabled** . Determines whether entry rebalancing is enabled. When rebalancing is enabled, the placement algorithm is consulted after modify and add operations, to determine whether the target entry should be moved to a different backend set.

- **rebalancing-scope** . Indicates which modified entries are candidates for rebalancing. A value of top-level indicates that only entries immediately below the entry-balancing base can be rebalanced. A value of any indicates that entries at any level below the entry-balancing base may be rebalanced.
- **rebalancing-minimum-percentage** . Specifies the minimum threshold for entries to be migrated from one backend set to a preferred backend set with a smaller size. Entries are not migrated unless the percentage difference between the value of the current backend set and the value of the preferred backend set exceeds this threshold. This parameter prevents unnecessarily migrating entries back and forth between backend sets of similar sizes.
- **rebalancing-subtree-size-limit** . Specifies the maximum size of a subtree that can be rebalanced.
- **poll-interval** . Specifies how long to wait between polling the size of the backends to determine how to rebalance; works in conjunction with the `rebalancing-minimum-percentage` property.
- **placement-criteria** . Determines which approach to use to select a destination backend for rebalancing. Possible values are: `entry-count`, `backend-size`, or `custom`.

The following figure illustrates an entry-balancing base DN and three subtrees, A, B, and C. If the rebalancing scope is set to `any`, any child entries under the base DN can be rebalanced. For example, if a change is made to entry A1, the entire subtree A might be rebalanced, depending upon how you have configured rebalancing. If the rebalancing scope is set to `top-level`, rebalancing is only triggered when entries at the top level, such as A, are modified. Changes made to subentries, such as A1 or A2, do not trigger rebalancing. Rebalancing is also triggered upon the addition of entries such as `A1,A2`, provided the scope is `any`.



MY TITLE Rebalancing at the Top Level

If you are writing your own 3rd party algorithm, you program dynamic rebalancing using the SelectRebalancingBackendSet method on the placement algorithm. For more information, see the Server SDK documentation.

Configuring dynamic rebalancing

About this task

This procedure describes how to configure dynamic rebalancing on an existing entry balancing configuration.

Steps

1. To configure entry rebalancing, you may create an entry counter placement algorithm, if the current placement algorithm does not support rebalancing. You can either do this using **`dsconfig`** in interactive mode, or using the **`dsconfig`** command line as follows:

```
$ dsconfig create-placement-algorithm \
  --processor-name dc_example_dc_com-eb-req-processor \
  --algorithm-name rebalancing --type entry-counter \
  --set enabled:true --set rebalancing-enabled:true
```

2. Remove any placement algorithm previously configured on this entry-balancing request processor.
3. You can throttle how many entries are being moved by the proxy, so that the backend servers do not have too heavy a load. To do this, set the rebalancing-queue-maximum-size property of the request processor created in the previous step. By default, it is set to 1000. If the load is too high, reduce this value as follows:

```
$ dsconfig set-request-processor-prop \
  --processor-name dc_example_dc_com-eb-req-processor \
  --set rebalancing-queue-maximum-size:50
```

4. Verify that the access logs are configured to display the subtrees being moved by dynamic rebalancing. The access logs provide a good way to monitor progress. So, if the write load on the backend servers is high and you see lots of rebalancing activity in the access log, lower the queue size to lower the rebalancing activity. You can configure the access log to display entry rebalancing processing information as follows:

```
$ dsconfig set-log-publisher-prop \
  --publisher-name "File-Based Access Logger" \
  --set log-entry-rebalancing-requests:true
```

About the move-subtree tool

The **`move-subtree`** tool allows you to specify subtrees for rebalancing. You specify the source server, the target server, and one or more base DNs identifying the subtrees you want to move. You can move small subtrees using the transactional method or move large subtrees, which does not use this method. Instead, the large subtree is not fully accessible during the move, so clients may get an "insufficient access rights error" if they try to access the subtree. As entries are moved, clients can read but not write to them. Once the transfer is complete, the entries are fully available to client requests.

This tool accepts a file containing a list of the base DNs of the subtrees you want to move.

ⓘ **Note:** The **`move-subtree`** tool requires users to have access to the extended operations and controls needed to run the tool. Make sure to apply the following ACIs to your data.

```
aci: (targetcontrol="1.3.6.1.4.1.30221.2.5.5 ||
      1.3.6.1.4.1.30221.2.5.24 || 1.3.6.1.4.1.30221.2.5.13")
     (version 3.0; acl "Allow admin to submit move-subtree controls";
      allow (read) userdn="ldap:///uid=admin,dc=example,dc=com";)
aci: (extop="1.3.6.1.4.1.30221.2.6.19")
     (version 3.0; acl "Allow admin to request move-subtree extended
      operation"; allow (read) userdn="ldap:///uid=admin,dc=example,dc=com";)
```

About the subtree-accessibility tool

The **`subtree-accessibility`** tool helps you determine if a subtree has restricted access and helps you fix any problems. If, during rebalancing, the Directory Server issues an alert that a subtree has been unavailable for too long, then you can use this tool to evaluate the problem. For example, if the **`move-subtree`** tool is interrupted by a host machine going down unexpectedly, the subtree might not be successfully moved. You can use the **`subtree-accessibility`** tool to evaluate and correct any problems with the subtrees, and then re-run the **`move-subtree`** tool.

Managing the global indexes in entry-balancing configurations

In an entry-balancing configuration, the Directory Proxy Server maintains the default RDN index as well as one or more in-memory global attribute indexes. The global indexes allow the Directory Proxy Server to select the correct backend server set for incoming operations, which avoids broadcasting operations to all backend sets.

The indexes may be preloaded from peer proxies or the backend directory servers when the server starts up, and are updated by certain operations that come through the Directory Proxy Server. For instance, when a new entry is added, the DN of the new entry is added to the DN index of the Directory Proxy Server performing the operation. The indexes are also fault-tolerant and can adapt to changes made in the backend servers without going through the Directory Proxy Server. For example, operations will be processed directly through the backend server.

This section describes when to create a global attribute index, how to reload the global index, how to monitor its growth, and how to prime the global index from a peer at start-up.

When to create a global attribute index

The RDN index is referenced whenever a modify, delete, or base search is requested. In other words, the RDN index is needed when the LDAP request contains the complete DN of the targeted entry. If the entry-balancing request processor is not configured to prime the rdn index at startup, then the index is populated over time as LDAP requests are processed.

A global attribute index is an optional index and is referenced when the Directory Proxy Server is handling a search request with an equality filter involving the attribute, such as the `telephoneNumber` attribute with the filter (telephoneNumber=+11234567890). Since the Directory Proxy Server does not know what the data within the subtree views looks like or how it will be searched, it cannot create or recommend default global attribute index definitions. The creation of a global attribute index is based on the range of equality-filtered search requests that the Directory Proxy Server will handle. The Directory Server must also have an equality or ordering index type for the associated attribute Local DB Index."

The common candidates for global attribute indexing are the uniquely-valued equality-indexed attributes on the external servers. Examples of these attributes are `uid`, `mail` and `telephoneNumber`. Though the values of the attribute need not be unique to be used as a global attribute index by the entry-balancing request processor.

Consider a Directory Proxy Server deployment that expects to handle frequent searches of the form "`(&(mail=user@example.com)(objectclass=person))`". Since the filter is constructed with an equality match and &-clause, we can use a global attribute index on the `mail` attribute to avoid forwarding the search request to each entry balanced dataset.

The following **`dsconfig`** command creates the global attribute index. Note that the mail attribute must be indexed for equality searches on each of the external servers behind the Directory Proxy Server.

```
$ bin/dsconfig create-global-attribute-index \
  --processor-name ou_people_dc_example_dc_com-eb-req-processor \
  --index-name mail --set prime-index:true \
```

After creating the index with **`dsconfig`**, the index will begin to be populated as search requests involving the mail attribute are made to the Directory Proxy Server. At this point, you can also use the **`reload-index`** tool to fully populate the index for optimal performance as described in the following section.

Reloading the global indexes

The Directory Proxy Server provides a tool, **`reload-index`**, which allows you to manually reload the Directory Proxy Server global indexes. You might need to reload the index when:

- The Directory Proxy Server fails to successfully load its global indexes on startup.
- Changes are made to the set of indexed attributes.
- Significant changes are made to the content in backend servers.
- The integrity of the index is in question.

You can use the tool to reload all configured indexes in the global index, including the RDN index and all attribute indexes, or to reload only those indexes you specify.

The tool schedules an operation to run within the Directory Proxy Server's process. You must supply LDAP connection information so that the tool can communicate with the server through its task interface. Tasks

can be scheduled to run immediately or at a later scheduled time. Once scheduled, you can manage the tasks using the **`manage-tasks`** tool.

Reloading all of the index

Steps

- Run the **`reload-index`** tool to reload all of the indexes within the scope of the `dc=example,dc=com` base DN. The task is performed as `cn=Directory Manager` on port 389 of the localhost server. The existing index contents are erased before reloading.

```
$ bin/reload-index --task --bindPassword password --baseDN
 "dc=example,dc=com"
```

Reloading the RDN and UID index

Steps

- To reload the RDN and UID index in the background so that the existing contents of these indexes can continue to be used, run the command as follows:

```
$ bin/reload-index --task --bindPassword password \
  --baseDN "dc=example,dc=com" --index rdn --index uid --background
```

Priming the backend server using the --fromDS option

About this task

You can force the Directory Proxy Server to prime from the backend directory server only using the `--fromDS` option, overriding the configuration of the `prime-index-source` property. You can do this on a one off basis if the global index appears to be growing too large. For example, run the command as follows:

Steps

- Run the **`reload-index`** command with the `--fromDS` option to prime the backend server.

```
$ bin/reload-index --bindPassword password --baseDN "dc=example,dc=com" --
fromDS
```

Monitoring the size of the global indexes

Over time, stale entries can build up in the global indexes because proxies do not communicate changes to the indexes with one another. The Directory Proxy Server continues to operate normally in this situation since the global indexes are only ever used as a hint at where to find entries.

The rate of this growth is typically very slow since in most environments the key attributes change infrequently. The global indexes themselves are also very compact. However, if the global indexes start to fill up the allocated memory, you may need to flush and reload them. The size of the global indexes can be monitored over LDAP using the following command:

```
$ bin/ldapsearch -b "cn=monitor" -D "uid=admin,dc=example,dc=com" -w password
 \
  "(objectClass=ds-entry-balancing-request-processor-monitor-entry)" \
  global-index-current-memory-percent
```

If the global indexes fill up, the Directory Proxy Server will continue to operate normally, but it will need to start evicting entries from the indexes, which will lead to more broadcast searches, reducing the overall throughput of the Directory Proxy Server.

To reload the indexes so that they no longer hold stale information, run the **`reload-index`** command with the `--fromDS` option so that data is loaded from backend directory servers. We recommend that you

reload the indexes during off-peak hours because it may have an impact on performance while the reload is in progress.

Sizing the global indexes

About this task

The Directory Proxy Server includes a tool, **`global-index-size`**, to help you estimate the size in memory of your global indexes. You can estimate the size of more than one index in a single invocation by providing multiple sets of options. The tool creates its estimate using the following information:

- **Number of keys in the index**. For example, for the built-in RDN index, the number of keys is the total number of entries in the Directory Server that are immediately below the balancing point. Entries more than one level below the balancing point, as well as entries that are not subordinate to the balancing point, will not be contained in the RDN index. For attribute indexes, the number of keys will be the number of unique values for that attribute in the entry-balanced portion of the data.
- **Average size of each key, in bytes**. For attributes indexes, the key is simply the attribute value. For the built-in RDN index, the key is the RDN directly below the balancing base DN. For example, for the DN `uid=user.0,dc=example,dc=com` under the balancing base DN of `dc=example,dc=com`, the key size is 10 bytes (the number of bytes in the RDN uid=user.0).
- **Estimated number of keys**. This value corresponds to the maximum number of keys you expect in your Directory Server. The number of keys is provided in the index-size configuration property of the `global-attribute-index` object when you configure an attribute index. For the built-in RDN index, the configured number of keys is provided in the `rdn-index-size` property. If you do not provide a value, the tool assumes that the configured number of keys is the same as the actual number of keys.

Steps

- Run the **`global-index-size`** to estimate the size of two separate indexes, both with 10,000,000 keys but with differing average key sizes. The configured number of keys is assumed to be equal to the actual number of keys:

```
$ bin/global-index-size --numKeys 10000000 \
  --averageKeySize 11 --numKeys 10000000 \
  --averageKeySize 15

Num Keys : Cfg. Num Keys : Avg. Key Size : Est. Memory Size
---------:---------------:---------------:-----------------
10000000 : 10000000      : 11            : 159 mb
10000000 : 10000000      : 15            : 197 mb
```

Priming the global indexes on startup

The Directory Proxy Server can prime the global indexes on startup from the backend directory server or from a peer proxy server, preferably one that resides on the same LAN or subnet. When priming occurs locally, you can avoid WAN bandwidth consumption and reduce the processing load on the directory servers in the topology. You can specify the data sources for the index priming and the order in which priming from these sources occurs.

Use the `prime-index-source` property to specify the sources of data, either `ds`, `file` or some combination of the two. The order you specify is the order in which priming from these sources will be attempted. For example, if you specify `prime-index-source:file,ds`, priming will be performed from the `global-index` data file created from the previous run of the directory servers. With the `file,ds` configuration, the contents of the global index are written to disk periodically if, and only if, the entire global index has been primed previously from a directory servers source either from startup or reloaded-index. Priming is most efficient if the source server is on the same local network as the Directory Proxy Server.

Configuring all indexes at startup

About this task

The following example configures the entry-balancing request processor so that it primes the global index from the persisted file, if present, or from an external directory servers source if necessary.

Steps

- Run the **dsconfig** tool to prime all indexes at startup.

```
$ bin/dsconfig set-request-processor-prop \
  --processor-name dc_example_dc_com-eb-req-processor \
  --set prime-all-indexes:true --set prime-index-source:file \
  --set prime-index-source:ds
```

Configuring the global indexes manually

About this task

If you do not want to configure priming during setup, you can configure index priming manually by creating an external server, creating a global attribute index, and then changing the entry-balancing request processor to load indexes from this external server.

Steps

1. Use the **dsconfig** tool to create an external server of the type PingDirectoryProxy Server to represent a peer of the Directory Proxy Server.

```
$ bin/dsconfig create-external-server \
  --server-name intra-proxy-host.example.com:3389 \
  --type PingDirectoryProxy-server \
  --set server-host-name:intra-proxy-host \
  --set server-port:338 \
  --set "bind-dn:cn=Directory Manager" \
  --set "password:secret123"
```

2. Create a global attribute index on the `uid` attribute as follows:

```
$ bin/dsconfig create-global-attribute-index \
  --processor-name dc_example+dc+com-eb-req-processor \
  --index-name uid \
```

3. Change the entry-balancing request processor to load the indexes at startup from the peer Directory Proxy Server using **dsconfig set-request-processor-prop** as described above.

Persisting the global index from a file

About this task

The PingDirectoryProxy Server supports periodically persisting the global index to a file and priming the global index from the persisted file when the server is restarted.

An Entry Balancing Request Processor can be configured to periodically persist the global index to disk, so that when the Entry Balancing Request Processor is reinitialized (on startup), it can prime the values from disk instead of putting load on the remote servers. Being able to read the index from disk eliminates the load on backend Directory Server instances if many PingDirectoryProxy Server instances were to come up at once.

An entry-balancing request processor can be configured to persist the global index to disk by including **file** as one of the prime index sources (with the **prime-index-source** property). The frequency at which the file is written is controlled by the **persist-global-index-frequency** property.

The global index needs to be fully primed before it will be persisted. It can be initially primed using a peer PingDirectoryProxy Server or from a backend Directory Server. On a running PingDirectoryProxy Server, when new global attribute indexes are added, the global index can be primed with those attribute indexes by running the **`rebuild-index`** tool. The **`rebuild-index`** tool always uses a remote server for priming the global index even if **`file`** is configured as a source). On subsequent restarts of the PingDirectoryProxy Server, the global index will be primed from the persisted file instead of going over the network to a remote server, which allows it to be primed much faster than if it were using a remote priming source. Also, during server startup, the global index priming works by using each configured **`prime-index-source`** property in the specified order until it is fully primed to take advantage of what is available locally before contacting one or more remote servers.

Steps

- The following **`dsconfig`** command prime all indexes at startup from a file.

```
dsconfig -n set-request-processor-prop \
  --processor-name entry-balancing \
  --set prime-index-source:file \
  --set prime-index-source:ds \
  --set persist-global-index-frequency:10s \
  --set persist-global-index-directory:/servers/proxy-1/index-files \
  --set prime-all-indexes:true
```

Priming or reloading the global indexes from Sun Directory servers

When priming or reloading a global index based on a Sun Directory Server environment, the Sun servers may become overwhelmed and unresponsive because of their method of streaming data. To reduce the impact of priming on these server, you can use the `prime-search-entry-per-second` property. To reduce the impact of reloading these indexes, use the `--searchEntryPerSecond` property of the `reload-index` command. These properties control the rate at which the Directory Proxy Server accepts search result entries from the backend directory servers.

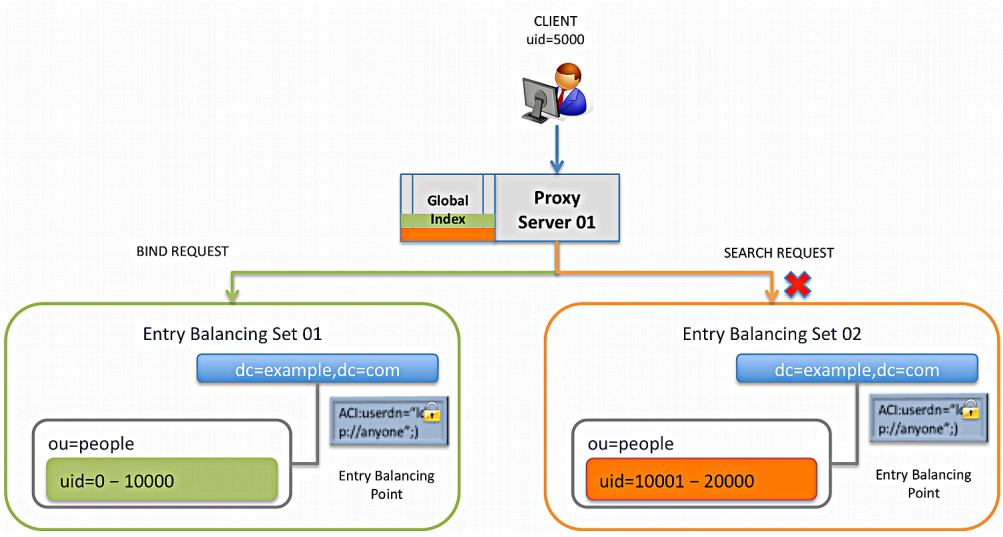
The `index-priming-idle-listener-timeout` property specifies the amount of time an extended operation response listener can be idle while the global index priming is in progress. For example, if the global index is priming and a backend server stops returning results but does not disconnect, this timeout can be used to force a retry of the operation.

To find the optimum rate, start low and specify a few thousand search entries per second. Then increase as necessary.

Working with alternate authorization identities

Access control rules in an entry-balanced deployment are configured in the Directory Server backend servers and require access to the entry contents of the user *issuing* the request. This can introduce a possible issue when clients to the Directory Proxy Server authenticate as users whose entries are among the entry-balanced sets. If the server which is processing a request does not contain the issuing user's entry, then the access control cannot be evaluated.

For example, consider a deployment that has two entry-balancing sets, set-01 and set-02. Set-01 has entries in the range `uid=0-10000`, while set-02 has entries for `uid=10001-20000`. The client with `uid=5000` binds to the Directory Proxy Server, which sends a BIND request to entry-balancing set-01. Next, the client sends a SEARCH request with filter "`(uid=15000)`". The Directory Proxy Server determines that `uid=15000` lives on entry-balancing set-02. The Directory Proxy Server then determines that the entry for the authenticated user with `uid=5000` does not exist in set-02 and that the access control handler would reject the SEARCH request issued by an unknown user.



MY TITLE Entry-Balancing Issue with Clients Not Present in the Underlying Data Set

One solution to this problem is to make use of an *alternate authorization identity* for the user, which references an entry that exists in all Directory Servers in all backend sets and has an equivalent set of access control rights as the authenticated user. The alternate authorization identity is used when the Directory Proxy Server observes that the Directory Server processing a request does not contain the entry of the user issuing the request.

The following sections cover the procedures to configure the alternate authorization identities for the Directory Proxy Server.

About alternate authorization identities

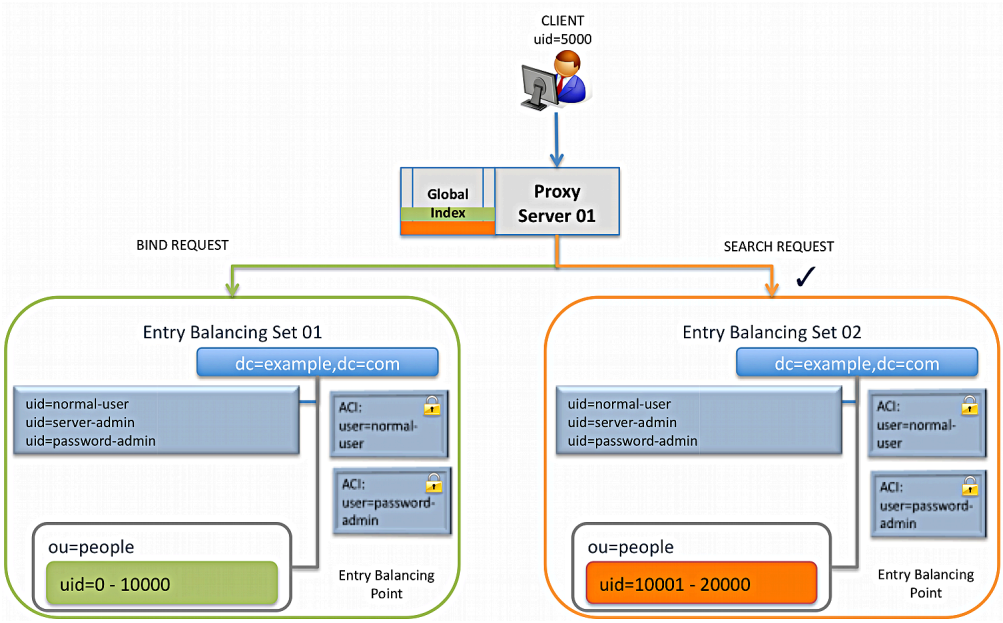
Whenever the Directory Proxy Server forwards a request to the backend set containing the user's entry, it forwards the request with an authorization identity that reflects the user's actual identity, since servers in that set already know about that user. However, when forwarding a request to a backend set that does *not* contain the user's entry, the Directory Proxy Server uses an *alternate authorization identity* that reflects the generic user with the same set of rights as the actual user issuing the request. Alternate authorization identities allow for the proper evaluation of access control rules for users whose entries are not present within an entry-balanced dataset.

There are typically only a few different generic class of users from an access control perspective, which can be placed in a portion of the DIT that is not below the entry-balancing base DN and is replicated to all servers in the topology. For example, assume that you have three classes of users: full administrators, password administrators, and normal users. You could create the following entries in the topology and assign them the appropriate access rights:

- uid=normal user,dc=example,dc=com
- uid=server-admin,dc=example,dc=com
- uid=password-admin,dc=example,dc=com

Returning to the example scenario, the client with `uid=5000` binds to the Directory Proxy Server, which sends a BIND request to entry-balancing set-01. Next, the client sends a SEARCH request for `uid=15000`. The Directory Proxy Server determines that `uid=15000` lives on entry-balancing set-02. Next, the Directory Proxy Server then determines that the client `uid=5000` does not have an entry on entry-balancing set-02. The Directory Proxy Server uses an *alternate authorization identity* that reflects the generic user, `uid=normal user`, which has the same set of rights as the client `uid=5000` who is issuing the request. The access control is accepted and the SEARCH request returns a response for `uid=5000`.

Whenever a user authenticates to the Directory Proxy Server, the server can keep track of which backend set holds that user's entry and determine whether an alternate authorization identity is required. The server can also determine which of these generic accounts best describes the rights that the user should have.



MY TITLE Alternate Authorization Identity Solves Access Control Issues in Entry-Balancing Deployments

When an alternate authorization identity is invoked, you will see `authzID='dn:uid=normal user,dc=example,dc=com'` in the server log, indicating that the alternate authorization identity was used. For example, if the `user.15000` is in a different backend set from `user.5000`, the log will show the following:

```
% bin/ldapsearch -D "uid=user.5000,ou=people,dc=example,dc=com" -w pasword \
  -b uid=user15000,ou=people,dc=example,dc=com "(objectclass=*)"

[18/Aug/2013:11:54:35 -0500] SEARCH REQUEST conn=153 op=1 msgID=2
via="app='Directory-Proxy address='127.0.0.1'
authzID='dn:uid=normal user,dc=example,dcom' sessionID='conn=2'
requestID='op=1'" base="uid=user.150000,ou=people,dc=example,dc=com"scope=2
filter="(objectclass=*)" attrs="ALL"

[18/Aug/2013:11:54:35 -0500] SEARCH REQUEST conn=153 op=1 msgID=2 resultCode=0
 etime=2.038
entriesReturned=1 authzDN="uid=normal-user,dc=example,dc=com"
```

Configuring alternate authorization identities

About this task

Alternate authorization identities are specified by the `authz-attribute` property of the entry-balancing request processor configuration object. By default, the `authz-attribute` property has the default value of `ds-authz-map-to-dn`, which is an attribute reserved for this purpose.

If a user entry has a value for `ds-authz-map-to-dn` whether it's explicitly contained in the entry or only present via a virtual attribute, then that will be used to specify the alternate authorization identity for the user. Otherwise, the default authorization identity (as indicated via the `authz-dn` configuration property) will be used to determine the alternate authorization identity.

Steps

1. Use **`dsconfig`** to set the `authz-dn` property of the entry-balancing request processor configuration. If any user among the balanced entries does not have an alternate authorization identity defined, the

Directory Proxy Server will use the value of the `authz-dn` property of the entry-balancing request processor configuration.

```
$ bin/dsconfig set-request-processor-prop \
  --processor-name dc_example_dc_com-eb-req-processor \
  --set "authz-dn:uid=normal user,dc=example,dc=com"
```

2. Create an auxiliary object class containing `ds-authz-map-to-dn` as an allowed attribute.
3. Add the auxiliary object class value to all user entries of interest.
4. Then, add the following attribute value to a `server-admin` user.

```
ds-authz-map-to-dn: uid=server-admin,dc=example,dc=com
```

Managing Entry-Balancing Replication

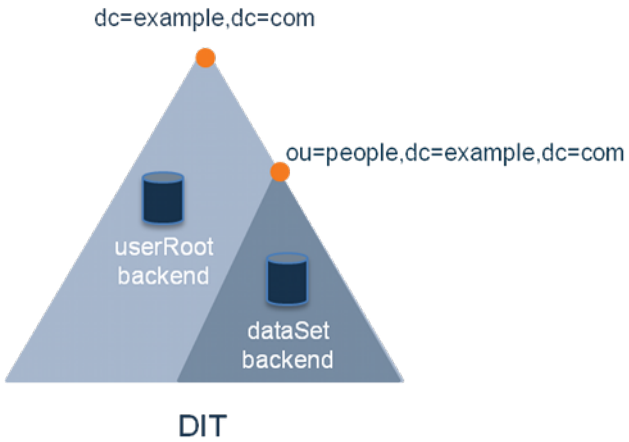
Replication in the PingDirectoryProxy Server synchronizes directory data between all servers in the topology. In a deployment using the entry-balancing feature, however, directory data under the entry-balancing point is split into multiple data sets. Each data set is replicated to ensure high availability between a subset of the servers in the topology. Other directory data, such as the schema or data above the entry-balancing point, is replicated between all servers in the topology.

This chapter presents the following information about replication in an entry-balancing environment:

Overview of replication in an entry-balancing environment

In an entry-balanced deployment, some data is replicated everywhere, such as the schema, the server registry, and other shared data, and some data is replicated only on certain servers. A replication domain contains all of the servers in a replicated topology and shares a schema. The replication domain is associated with the base DN and must be a base DN of a backend.

By default, replication propagates updates to all replication servers in the topology. Updates to data under the entry-balancing point, however, must be replicated only among server instances in the same data set. Replication requires that, in such deployments, the Directory Server is configured with a replication set name global configuration property, and two backends. One backend has a base DN that is replicated globally (such as `dc=example,dc=com`) and the second backend has a base DN associated with the entry-balancing point (such as `ou=people,dc=example,dc=com`).



MY TITLE Global and Restricted Backends

If a data set name is not defined when you set up the Directory Proxy Server, one will be provided by default. The proper configuration of an entry-balancing environment requires coordination between the Directory Server and Directory Proxy Server. Once replication is enabled, the replication domain may be designated as the domain participating in entry balancing.

Review the *Directory Server Administration Guide* for more details about replication, managing the replication topology, and working with multiple backends.

Replication prerequisites in an entry-balancing deployment

Replication in an entry-balanced deployment requires the following:

- **Multiple local DB backends**. When you set up the Directory Server instances, you need two backends, a global backend for globally replicated data, such as `userRoot`, and a backend for the balancing point base DN, `dataSet`. Both backends need to be enabled for replication and initialized separately.
- **Replication set name**. Every Directory Server in your replicated topology must have a replication set name. This replication set name coordinates the Directory Proxy Server and the Directory Server. The restricted domain is only replicated within instances using the same replication set name.
- **Multiple Directory Proxy Server subtree views**. The entry-balanced proxy configuration relies on multiple subtree views, one for the globally replicated base DN and one for the entry-balancing point base DN. The globally replicated base DN will have a proxying request processor associated with it. The restricted base DN will have an entry-balancing request processor associated with it. This configuration is best achieved using the **`create-initial-proxy-config`** tool after running setup.

About the --restricted argument of the dsreplication command-line Tool

When enabling replication for a server that takes part in an entry balanced environment, it is recommended that the multiple domains involved are enabled at the same time. There is a global domain, and a restricted domain, where the restricted domain represents the entry-balancing point. Each base DN is defined in a separate Local DB Backend. The **`dsreplication`** CLI tool has a `--restricted` argument that is used to specify which base DN is considered an entry-balancing point.

Using the --restricted argument of the dsreplication command-line tool

Steps

- Run **`dsreplication`** to enable replication between two servers with entry balancing.
  - You can run the command in non-interactive mode as follows:

```
$ bin/dsreplication enable --host1 host1.example.com \
  --port1 1389 --bindDN1 "cn=Directory Manager" \
  --bindPassword1 secret --replicationPort1 8989 \
  --host2 host2.example.com --port2 2389 \
  --bindDN2 "cn=Directory Manager" --bindPassword2 secret \
  --replicationPort2 8989 --baseDN dc=example,dc=com \
  --baseDN ou=people,dc=example,dc=com \
  --restricted ou=people,dc=example,dc=com
```

  - Alternatively, you can enable replication using the interactive command line, making sure to specify that an entry balancing is being used and specifying the base DN of the entry-balancing point. After entering **`dsreplication`** and entering the LDAP connection parameters, follow the prompts presented.

```
You must choose at least one base DN to be replicated.

Replicate base DN dc=example,dc=com? (yes / no) [yes]: yes

Replicate base DN ou=people,dc=example,dc=com? (yes / no) [yes]: yes

Do you plan to configure entry balancing using the Directory Proxy Server?
 (yes / no) [no]: yes

Is dc=example,dc=com an entry-balancing point? (yes / no) [no]: no
```

```
Is ou=people,dc=example,dc=com an entry-balancing point? (yes / no) [no]:
 yes
```

Checking the status of replication in an entry-balancing deployment

About this task

You can use the **dsreplication** status tool to check the status of an entry-balancing deployment. In this example, the `ou=people,dc=example,dc=com` subtree is entry-balanced. The data is split into two sets, `set1` and `set2`. The servers `host1` and `host2` are in replication set `set1` and servers `host3` and `host4` are in replication set `set2`.

Steps

- Run the **dsreplication** command to get a status of replication in the entry-balancing deployment. To view a specific set, use the `--setName` option to see only the specific replication set; otherwise, all of the sets will be displayed by default.

```
$ bin/dsreplication status --hostname host1.example.com \
  --port 1389 --adminUID admin --adminPassword secret
```

```
          --- Replication Status for dc=example,dc=com: Enabled ---

Server    : Entries: Replication Backlog: Oldest Backlog Change Age : Port :Security

----------:--------:--------------------:---------------------------:------:--------

austin1.example.com:1389 : 1000    : 0            : N/A             : 8989 : Enabled
austin2.example.com:2389 : 1000    : 0            : N/A             : 8989 : Enabled
newyork1.example.com:3389: 1000    : 0            : N/A             : 8989 : Enabled
newyork2.example.com:4389: 1000    : 0            : N/A             : 8989 : Enabled


 -- Replication Status for ou=people,dc= example,dc=com (Set: set1): Enabled --

Server    : Entries: Replication Backlog: Oldest Backlog Change Age : Port :Security

----------:--------:--------------------:---------------------------:------:--------

austin1.example.com:1389 : 1000000 : 0            : N/A            : 8989 : Enabled
austin2.example.com:2389 : 1000000 : 0            : N/A            : 8989 : Enabled


---Replication Status for ou=people,dc= example,dc=com (Set: set2): Enabled ---

Server    : Entries: Replication Backlog: Oldest Backlog Change Age : Port :Security

----------:--------:--------------------:---------------------------:------:--------

newyork1.example.com:3389 : 1000000 : 0           : N/A             : 8989 : Enabled
newyork2.example.com:4389 : 1000000 : 0           : N/A             : 8989 : Enabled
```

Example of configuring entry-balancing replication

This section describes how to set up a four-server replication topology that uses entry balancing to distribute entries across the servers. The procedure assumes that none of the servers have participated in any previous replication topology. This is supported for one or multiple entry balancing domains.

Assumptions

The example uses the LDAP (389) and replication (8989) ports respectively. It configures the following hosts:

- austin1.example.com
- newyork1.example.com
- austin2.example.com
- newyork2.example.com

In this example, we have a global domain of `dc=example,dc=com`, which is replicated across all servers. The data below the entry-balancing point of `ou=people,dc=example,dc=com` is distributed across two

data sets, dataSet1 and dataSet2. Each data set is replicated between two directory servers. Each of these servers is associated with one of two locations, Austin and New York.

Configuration summary

To configure replication in an entry-balanced deployment, you must do the following:

- Install two directory servers in an Austin location and two directory servers in a New York location.
- Create a new backend, called **`dataset`**, to store the entry-balancing data set.
- Define entry-balancing set names **`dataSet1`** and **`dataSet2`** for assignment to the **`replication-set-name`** Global Configuration Property of the Directory Server instances.
- Import the data representing the global domain, stored in userRoot, into a server. Choose a server for each of the entry-balancing data sets, both stored in the backend named **`dataset`**.
- Enable replication and initialize remaining servers.
- Configure the proxies.
- Check the status of replication.

Installing the Directory Server

About this task
First, install the Directory Server instances. In this example, we install the following four servers, two in the Austin location and two in the New York location:

- austin1.example.com
- austin2.example.com
- newyork1.example.com
- newyork2.example.com

Steps

1. We install the first server, austin1, as follows:

```
root@austin1# ./setup --cli --baseDN dc=example,dc=com \
--ldapPort 389 --rootUserDN "cn=Directory Manager" \
--rootUserPassword pass --no-prompt --acceptLicense
```

2. Install the second Austin server, austin2, in the same way:

```
root@austin2 # ./setup --cli --baseDN dc=example,dc=com \
--ldapPort 389 --rootUserDN "cn=Directory Manager" \
--rootUserPassword pass --no-prompt --acceptLicense
```

3. Next, install the two New York servers, newyork1 and newyork2, as follows:

```
root@newyork1# ./setup --cli --baseDN dc=example,dc=com \
--ldapPort 389 --rootUserDN "cn=Directory Manager" \
--rootUserPassword pass --no-prompt --acceptLicense

root@newyork# ./setup --cli --baseDN dc=example,dc=com \
--ldapPort 389 --rootUserDN "cn=Directory Manager" \
--rootUserPassword pass --no-prompt --acceptLicense
```

Creating the database backends and defining the replication set name

Steps

1. On all servers, create the dataset backend as follows:

```
./bin/dsconfig --no-prompt create-backend \
--backend-name dataset --type local-db --set enabled:true \
--set base-dn:ou=people,dc=example,dc=com
```

2. Set the replication set name for `austin1.example.com` and `newyork1.example.com` to dataset1:

```
./bin/dsconfig --no-prompt \
set-global-configuration-prop \
--set replication-set-name:dataset1
```

3. Set the replication set name for `austin2.example.com` and `newyork1.example.com`to dataset2:

```
./bin/dsconfig --no-prompt \
set-global-configuration-prop \
--set replication-set-name:dataset2
```

Creating and setting the locations

Steps

1. On the Austin servers, create the two locations, newyork and austin, and set the location of this instance to austin:

```
./bin/dsconfig --no-prompt create-location --location-name austin

./bin/dsconfig --no-prompt create-location --location-name newyork \
--set preferred-failover-location:austin

./bin/dsconfig --no-prompt set-location-prop --location-name austin \
--add preferred-failover-location:newyork

./bin/dsconfig --no-prompt set-global-configuration-prop \
--set location:austin
```

2. For the New York servers, set the location to newyork:

```
./bin/dsconfig --no-prompt create-location \
--location-name austin

./bin/dsconfig --no-prompt create-location \
--location-name newyork \
--set preferred-failover-location:austin

./bin/dsconfig --no-prompt set-location-prop \
--location-name austin \
--add preferred-failover-location:newyork

./bin/dsconfig --no-prompt set-global-configuration-prop \
--set location:newyork
```

Importing the entries

About this task

We import the userRoot data, based on data defined in the `userRoot.ldif` file, into one server. This file does not contain entries at or within the entry-balancing point, `ou=people,dc=example,dc=com`.

Steps

1. Use the **`import-ldif`** command to import the userRoot data.

```
root@austin1# ./bin/import-ldif --backendID userRoot \
--ldifFile /data/userRoot.ldif \
--includeBranch dc=example,dc=com \
--rejectFile /data/austin1-import-rejects \
--port 389
--hostname austin1.example.com
```

2. Import the dataSet1 data on one server into the dataset backend, which is assigned the dataSet1 **replication-set-name**.

```
root@austin1# ./bin/import-ldif --backendID dataset \
--ldifFile /data/dataset1.ldif \
--includeBranch ou=people,dc=example,dc=com \
--rejectFile /data/austin1-dataset-import-rejects \
--hostname austin1.example.com --port 389
```

3. Import the dataSet2 data on one server into the dataset backend, which is assigned the dataSet2 **replication-set-name**.

```
root@austin2# ./bin/import-ldif --backendID dataset \
--ldifFile /data/dataset2.ldif \
--includeBranch ou=people,dc=example,dc=com \
--rejectFile /data/austin2-dataset-import-rejects \
--hostname austin2.example.com --port 389
```

Enabling replication in an entry-balancing deployment

About this task

Now we can enable replication between the servers and initialize the remaining servers without data. Notice that we specify the `--restricted` domain in the **dsreplication** command.

Steps

1. Run **dsreplication enable** to enable the servers in the topology. The first invocation of this command creates the admin account.

```
root@austin1# ./bin/dsreplication enable \
--host1 austin1.example.com \
--port1 389 --bindDN1 "cn=directory manager" \
--bindPassword1 pass --host2 austin2.example.com \
--port2 389 --bindDN2 "cn=directory manager" \
--bindPassword2 pass \
--replicationPort1 8989 \
--replicationPort2 8989 \
--baseDN dc=example,dc=com \
--baseDN ou=people,dc=example,dc=com \
--restricted ou=people,dc=example,dc=com \
--adminUID admin --adminPassword pass --trustAll \
--no-prompt
```

2. Enable replication between austin1 and newyork1. This procedure automatically enables replication between austin2 and newyork1 as well.

```
root@austin1# ./bin/dsreplication enable \
--host1 austin1.example.com \
--port1 389 --bindDN1 "cn=directory manager" \
--bindPassword1 pass --host2 newyork1.example.com \
--port2 389 --bindDN2 "cn=directory manager" \
--bindPassword2 pass \
--replicationPort1 8989 \
--replicationPort2 8989 \
--baseDN dc=example,dc=com \
--baseDN ou=people,dc=example,dc=com \
--restricted ou=people,dc=example,dc=com \
--adminUID admin --adminPassword pass --trustAll \
--no-prompt
```

3. Enable replication between austin1 and newyork2. This will complete the entry-balancing replication setup.

```
root@austin1# ./bin/dsreplication enable \
  --host1 austin1.example.com \
  --port1 389 --bindDN1 "cn=directory manager" \
  --bindPassword1 pass --host2 newyork2.example.com \
  --port2 389 --bindDN2 "cn=directory manager" \
  --bindPassword2 pass \
  --replicationPort1 8989 \
  --replicationPort2 8989 \
  --baseDN dc=example,dc=com \
  --baseDN ou=people,dc=example,dc=com \
  --restricted ou=people,dc=example,dc=com \
  --adminUID admin --adminPassword pass --trustAll \
  --no-prompt
```

4. Initialize the remaining servers without data. The global domain, **`dc=example,dc=com`** needs to be initialized on austin2, newyork1 and newyork2. The **`ou=people,dc=example,dc=com`** entry-balancing domain needs to be initialized from austin1 to newyork2, and then again from austin2 to newyork2. We will combine these steps by initializing both domains with one invocation once austin2 is initialized with the global domain.

```
root@austin1# ./bin/dsreplication initialize \
  --hostSource austin1.example.com --portSource 389 \
  --hostDestination austin2.example.com \
  --portDestination 389 --adminUID admin \
  --adminPassword password \
  --baseDN dc=example,dc=com \
  --no-prompt

root@austin1# ./bin/dsreplication initialize \
  --hostSource austin1.example.com --portSource 389 \
  --hostDestination newyork1.example.com \
  --portDestination 389 --adminUID admin \
  --adminPassword password \
  --baseDN dc=example,dc=com \
  --baseDN ou=people,dc=example,dc=com \
  --no-prompt

root@austin2# ./bin/dsreplication initialize \
  --hostSource austin2.example.com --portSource 389 \
  --hostDestination newyork2.example.com \
  --portDestination 389 --adminUID admin \
  --adminPassword password \
  --baseDN dc=example,dc=com \
  --baseDN ou=people,dc=example,dc=com \
  --no-prompt
```

Checking the status of replication

About this task

Once replication has been configured, check the status of the replication topology using the **`dsreplication status`** command.

Steps

- Run the **`dsreplication status`** command to check its status.

```
root@austin1# ./bin/dsreplication status \
--adminPassword pass --no-prompt --port 389
```

Managing the Directory Proxy Server

Once you have configured the PingDirectoryProxy Server, you can manage the day-to-day operations of your deployment using the monitoring and logging features. This chapter provides procedures to help you configure logging and monitor your deployment.

This chapter includes the following sections:

Managing logs

The Directory Proxy Server provides a number of different types of log publishers that can be used to provide information about how the server is processing.

About the default logs

You can view all logs in the `PingDirectoryProxy/logs` directory. This section provides information about the following default logs:

- Error Log
- server.out Log
- Debug Log
- Config Audit Log and the Configuration Archive
- Access Log
- Setup Log
- Tool Log
- LDAP SDK Debug Log

Error log

By default, this log file is available at `logs/errors` below the server install root and it provides information about warnings, errors, and other significant events that occur within the server. A number of messages are written to this file on startup and shutdown, but while the server is running there is normally little information written to it. In the event that a problem does occur, however, the server writes information about that problem to this file.

The following is an example of a message that might be written to the error log:

```
[11/Apr/2011:10:31:53.783 -0500] category=CORE severity=NOTICE msgID=458887
 msg="The Directory Server has started successfully"
```

The category field provides information about the area of the server from which the message was generated. Available categories include:

ACCESS_CONTROL, ADMIN, ADMIN_TOOL, BACKEND, CONFIG, CORE, DSCONFIG, EXTENSIONS, PROTOCOL, SCHEMA, JEB, SYNC, LOG, PLUGIN, PROXY, QUICKSETUP, REPLICATION, RUNTIME_INFORMATION, TASK, THIRD_PARTY, TOOLS, USER_DEFINED, UTIL, VERSION.

The severity field provides information about how severe the server considers the problem to be. Available severities include:

- **DEBUG** – Used for messages that provide verbose debugging information and do not indicate any kind of problem. Note that this severity level is rarely used for error logging, as the Directory Proxy Server provides a separate debug logging facility as described below.
- **INFORMATION** – Used for informational messages that can be useful from time to time but are not normally something that administrators need to see.
- **MILD_WARNING** – Used for problems that the server detects, which can indicate something unusual occurred, but the warning does not prevent the server from completing the task it was working on. These warnings are not normally something that should be of concern to administrators.

- **MILD_ERROR** – Used for problems detected by the server that prevented it from completing some processing normally but that are not considered to be a significant problem requiring administrative action.
- **NOTICE** – Used for information messages about significant events that occur within the server and are considered important enough to warrant making available to administrators under normal conditions.
- **SEVERE_WARNING** – Used for problems that the server detects that might lead to bigger problems in the future and should be addressed by administrators.
- **SEVERE_ERROR** – Used for significant problems that have prevented the server from successfully completing processing and are considered important.
- **FATAL_ERROR** – Used for critical problems that arise which might leave the server unable to continue processing operations normally.

The messages written to the error log may be filtered based on their severities in two ways. First, the error log publisher has a `default-severity` property, which may be used to specify the severity of messages logged regardless of their category. By default, this includes the NOTICE, SEVERE_WARNING, SEVERE_ERROR, and FATAL_ERROR severities.

You can override these severities on a per-category basis using the `override-severity` property. If this property is used, then each value should consist of a category name followed by an equal sign and a comma-delimited set of severities that should be logged for messages in that category. For example, the following override severity would enable logging at all severity levels in the PROTOCOL category:

```
protocol=debug,information,mild-warning,mild-error,notice,severe-
warning,severe-error,fatal-error
```

Note that for the purposes of this configuration property, any underscores in category or severity names should be replaced with dashes. Also, severities are not inherently hierarchical, so enabling the DEBUG severity for a category will not automatically enable logging at the INFORMATION, MILD_WARNING, or MILD_ERROR severities.

The error log configuration may be altered on the fly using tools like **`dsconfig`**, the Administrative Console, or the LDIF connection handler, and changes will take effect immediately. You can configure multiple error logs that are active in the server at the same time, writing to different log files with different configurations. For example, a new error logger may be activated with a different set of default severities to debug a short-term problem, and then that logger may be removed once the problem is resolved, so that the normal error log does not contain any of the more verbose information.

server.out log

The `server.out` file holds any information written to standard output or standard error while the server is running. Normally, it includes a number of messages written at startup and shutdown, as well as information about any administrative alerts generated while the server is running. In most cases, this information is also written to the error log. The `server.out` file can also contain output generated by the JVM. For example, if garbage collection debugging is enabled, or if a stack trace is requested via "kill -QUIT" as described in a later section, then output is written to this file.

Debug log

The debug log provides a means of obtaining information that can be used for troubleshooting problems but is not necessary or desirable to have available while the server is functioning normally. As a result, the debug log is disabled by default, but it can be enabled and configured at any time.

Some of the most notable configuration properties for the debug log publisher include:

- **enabled** – Indicates whether debug logging is enabled. By default, it is disabled.
- **log-file** – Specifies the path to the file to be written. By default, debug messages are written to the `logs/debug` file.
- **default-debug-level** – Specifies the minimum log level for debug messages that should be written. The default value is “error,” which only provides information about errors that occur during processing (for example, exception stack traces). Other supported debug levels include warning, info, and verbose.

Note that unlike error log severities, the debug log levels are hierarchical. Configuring a specified debug level enables any debugging at any higher levels. For example, configuring the info debug level automatically enables the warning and error levels.

- **default-debug-category** – Specifies the categories for debug messages that should be written. Some of the most useful categories include caught (provides information and stack traces for any exceptions caught during processing), database-access (provides information about operations performed in the underlying database), protocol (provides information about ASN.1 and LDAP communication performed by the server), and data (provides information about raw data read from or written to clients).

As with the error and access logs, multiple debug loggers can be active in the server at any time with different configurations and log files to help isolate information that might be relevant to a particular problem.

ⓘ **Note:** Enabling one or more debug loggers can have a significant impact on server performance. We recommend that debug loggers be enabled only when necessary, and then be scoped so that only pertinent debug information is recorded.

Debug targets can be used to further pare down the set of messages generated. For example, you can specify that the debug logs be generated only within a specific class or package. If you need to enable the debug logger, you should work with your authorized support provider to best configure the debug target and interpret the output.

Audit log

The audit log is a specialized version of the access log, used for troubleshooting problems that may occur in the course of processing. The log records all changes to directory data in LDIF format so that administrators can quickly diagnose the changes an application made to the data or replay the changes to another server for testing purposes.

The audit log does not record authentication attempts but can be used in conjunction with the access log to troubleshoot security-related issues. The audit log is disabled by default because it does adversely impact the server's write performance.

By default, if you enable the audit log on the Directory Proxy Server, the `userPassword` and `authPassword` attribute values are obscured, Each value of an obscured attribute is replaced in the audit log with a string of the form `"***** OBSCURED VALUE *****"`. You can unobscure these attributes by deleting them from the `obscure-attribute` property.

Config audit log and the configuration archive

The configuration audit log provides a record of any changes made to the server configuration while the server is online. This information is written to the `logs/config-audit.log` file and provides information about the configuration change in the form that may be used to perform the operation in a non-interactive manner with the **`dsconfig`** command. Other information written for each change includes:

- Time that the configuration change was made.
- Connection ID and operation ID for the corresponding change, which can be used to correlate it with information in the access log.
- DN of the user requesting the configuration change and the method by which that user authenticated to the server.
- Source and destination addresses of the client connection.
- Command that can be used to undo the change and revert to the previous configuration for the associated configuration object.

In addition to information about the individual changes that are made to the configuration, the Directory Proxy Server maintains complete copies of all previous configurations. These configurations are provided in the `config/archived-configs` directory and are gzip-compressed copies of the `config/config.ldif` file in use before the configuration change was made. The file names contain timestamps that indicate when that configuration was first used.

Access and audit log

The access log provides information about operations processed within the server. The default access log file is written to `logs/access`, but multiple access loggers can be active at the same time, each writing to different log files and using different configurations.

By default, a single access log message is generated, which combines the elements of request, forward, and result messages. If an error is encountered while attempting to process the request, then one or more forward-failed messages may also be generated.

```
[01/Jun/2011:11:10:19.692 -0500] CONNECT conn=49 from="127.0.0.1"
 to="127.0.0.1"
  protocol="LDAP+TLS" clientConnectionPolicy="default"
[01/Jun/2011:11:10:19.764 -0500] BIND RESULT conn=49 op=0 msgID=1 version="3"
  dn="cn=Directory Manager" authType="SIMPLE" resultCode=0 etime=0.401
  authDN="cn=Directory Manager,cn=Root DNs,cn=config"
 clientConnectionPolicy="default"
[01/Jun/2011:11:10:19.769 -0500] SEARCH RESULT conn=49 op=1 msgID=2
  base="ou=People,dc=example,dc=com" scope=2 filter="(uid=1)" attrs="ALL"
  resultCode=0 etime=0.549 entriesReturned=1
[01/Jun/2011:11:10:19.788 -0500] DISCONNECT conn=49 reason="Client Unbind"
```

Each log message includes a timestamp indicating when it was written, followed by the operation type, the connection ID (which is used for all operations processed on the same client connection), the operation ID (which can be used to correlate the request and response log messages for the operation), and the message ID used in LDAP messages for this operation.

The remaining content for access log messages varies based on the type of operation being processed, and whether it is a request or a result message. Request messages generally include the most pertinent information from the request, but generally omit information that is sensitive or not useful.

Result messages include a `resultCode` element that indicates whether the operation was successful or if failed and an `etime` element that indicates the length of time in milliseconds that the server spent processing the operation. Other elements that might be present include the following:

- **origin=replication** – Operation that was processed as a result of data synchronization (for example, replication) rather than a request received directly from a client.
- **message** – Text that was included in the `diagnosticMessage` field of the response sent to the client.
- **additionalInfo** – Additional information about the operation that was not included in the response sent back to the client.
- **authDN – DN** of the user that authenticated to the server (typically only included in bind result messages).
- **authzDN – DN** of an alternate authorization identify used when processing the operation (for example, if the proxied authorization control was included in the request).
- **authFailureID** – Unique identifier associated with the authentication failure reason (only included in non-successful bind result messages).
- **authFailureReason** – Information about the reason that a bind operation failed that might be useful to administrators but was not included in the response to the client for security reasons.
- **responseOID** – OID included in an extended response returned to the client.
- **entriesReturned** – Number of matching entries returned to the client for a search operation.
- **unindexed=true** – Indicates that the associated search operation could not be sufficiently processed using server indexes and a significant traversal through the database was required.

Note that this is not an exhaustive list, and elements that are not listed here may also be present in access log messages. The LDAP SDK for Java provides an API for parsing access log messages and provides access to all elements that they may contain.

The Directory Proxy Server provides a second access log implementation called the *audit log*, which is used to provide detailed information about write operations (add, delete, modify, and modify DN)

processed within the server. If the audit log is enabled, the entire content of the change is written to the audit log file (which defaults to `logs/audit`) in LDIF form.

The PingDirectoryProxy Server also provides a very rich classification system that can be used to filter the content for access log files. This can be helpful when debugging problems with client applications, because it can restrict log information to operations processed only by a particular application (for example, based on IP address and/or authentication DN), only failed operations, or only operations taking a long time to complete, etc.

Setup log

The **`setup`** tool writes a log file providing information about the processing that it performs. By default, this log file is written to `logs/setup.log` although a different name may be used if a file with that name already exists, because the **`setup`** tool has already been run. The full path to the setup log file is provided when the **`setup`** tool has completed.

Tool log

Many of the administrative tools provided with the Directory Proxy Server (for example, **`import-ldif`**, **`export-ldif`**, **`backup`**, **`restore`**, etc.) can take a significant length of time to complete write information to standard output or standard error or both while the tool is running. They also write additional output to files in the `logs/tools` directory (for example, **logs/tools/ import-ldif.log**). The information written to these log files can be useful for diagnosing problems encountered while they were running. When running via the server tasks interface, log messages generated while the task is running may alternately be written to the server error log file.

LDAP SDK debug log

This log can be used to help examine the communication between the Directory Server and the Directory Proxy Server. It contains information about exceptions that occur during processing, problems establishing and terminating network connections, and problems that occur during the reading and writing of LDAP messages and LDIF entries. You can configure the types of debugging that should be enabled, the debug level that should be used, and whether debug messages should include stack traces. As for other file-based loggers, you can also specify the rotation and retention policies.

Types of log publishers

The PingDirectoryProxy Server provides a number of differently types of loggers that can be used to get processing information about the server. There are three primary types of loggers:

- **Access loggers** provide information about operations processed within the server. They can be used for understanding the operations performed by clients and debugging problems with directory-enabled applications, and they can also be used for collecting usage information for performance and capacity planning purposes.
- **Error loggers** provide information about warnings, errors, or significant events that occur within the server.
- **Debug loggers** can provide detailed information about processing performed by the server, including any exceptions caught during processing, detailed information about data read from or written to clients, and accesses to the underlying database.

By default, the following log publishers are enabled on the system:

- File-based access logger
- File-based error logger
- Failed-operations access logger

The PingDirectoryProxy Server also provides the follow log publishers that are disabled by default:

- File-based debug logger
- File-based audit logger
- Expensive operations access logger

- Successful searches with no entries returned access logger

Creating new log publishers

The PingDirectoryProxy Server provides customization options to help you create your own log publishers with the **dsconfig** command.

When you create a new log publisher, you must also configure the log retention and rotation policies for each new publisher. For more information, see Configuring Log Rotation and Configuring Log Retention.

Creating a new log publisher

Steps

1. Use the **dsconfig** command in non-interactive mode to create and configure the new log publisher. This example shows how to create a logger that only logs disconnect operations.

```
$ bin/dsconfig create-log-publisher \
  --type file-based-access --publisher-name "Disconnect Logger" \
  --set enabled:true \
  --set "rotation-policy:24 Hours Time Limit Rotation Policy" \
  --set "rotation-policy:Size Limit Rotation Policy" \
  --set "retention-policy:File Count Retention Policy" \
  --set log-connects:false \
  --set log-requests:false --set log-results:false \
  --set log-file:logs/disconnect.log
```

> ⓘ **Note:** To configure compression on the logger, add the option to the previous command:
>
> ```
> --set compression-mechanism: gzip
> ```
>
> Compression cannot be disabled or turned off once configured for the logger. Therefore, careful planning is required to determine your logging requirements including log rotation and retention with regards to compressed logs.

2. If needed, view log publishers with the following command:

```
$ bin/dsconfig list-log-publishers
```

Creating a log publisher using dsconfig interactive command-line mode

Steps

1. On the command line, type **bin/dsconfig**.
2. Authenticate to the server by following the prompts.
3. On the main menu, select the option to configure the log publisher.
4. On the **Log Publisher** menu, select the option to create a new log publisher.
5. Select the Log Publisher type. In this case, select **File-Based Access Log Publisher**.
6. Type a name for the log publisher.
7. Enable it.
8. Type the path to the log file, relative to the Directory Proxy Server root. For example, `logs/disconnect.log`.
9. Select the rotation policy to use for this log publisher.
10. Select the retention policy to use for this log publisher.
11. On the Log Publisher Properties menu, select the option for `log-connects:false`, `log-disconnects:true`, `log-requests:false`, and `log-results:false`.

12. Type `f` to apply the changes.

About log compression

The Directory Proxy Server supports the ability to compress log files as they are written. This feature can significantly increase the amount of data that can be stored in a given amount of space, so that log information can be kept for a longer period of time.

Because of the inherent problems with mixing compressed and uncompressed data, compression can only be enabled at the time the logger is created. Compression cannot be turned on or off once the logger is configured. Further, because of problems in trying to append to an existing compressed file, if the server encounters an existing log file at startup, it will rotate that file and begin a new one rather than attempting to append to the previous file.

Compression is performed using the standard gzip algorithm, so compressed log files can be accessed using readily-available tools. The **`summarize-access-log`** tool can also work directly on compressed log files, rather than requiring them to be uncompressed first. However, because it can be useful to have a small amount of uncompressed log data available for troubleshooting purposes, administrators using compressed logging may wish to have a second logger defined that does not use compression and has rotation and retention policies that will minimize the amount of space consumed by those logs, while still making them useful for diagnostic purposes without the need to uncompress the files before examining them.

You can configure compression by setting the `compression-mechanism` property to have the value of "gzip" when creating a new logger.

About log signing

The Directory Proxy Server supports the ability to cryptographically sign a log to ensure that it has not been modified in any way. For example, financial institutions require audit logs for all transactions to check for correctness. Tamper-proof files are therefore needed to ensure that these transactions can be properly validated and ensure that they have not been modified by any third-party entity or internally by unscrupulous employees. You can use the **`dsconfig`** tool to enable the `sign-log` property on a Log Publisher to turn on cryptographic signing.

When enabling signing for a logger that already exists and was enabled without signing, the first log file will not be completely verifiable because it still contains unsigned content from before signing was enabled. Only log files whose entire content was written with signing enabled will be considered completely valid. For the same reason, if a log file is still open for writing, then signature validation will not indicate that the log is completely valid because the log will not include the necessary "end signed content" indicator at the end of the file.

To validate log file signatures, use the **`validate-file-signature`** tool provided in the `bin` directory of the server (or the `bat` directory for Windows systems).

Once you have enabled this property, you must disable and then re-enable the Log Publisher for the changes to take effect.

About encrypting log files

The Directory Proxy Server supports the ability to encrypt log files as they are written. The **`encrypt-log`** configuration property controls whether encryption will be enabled for the logger. Enabling encryption causes the log file to have an **`.encrypted`** extension (and if both encryption and compression are enabled, the extension will be **`.gz.encrypted`**). Any change that affects the name used for the log file could prevent older files from getting properly cleaned up.

Like compression, encryption can only be enabled when the logger is created. Encryption cannot be turned on or off once the logger is configured. For any log file that is encrypted, enabling compression is also recommended to reduce the amount of data that needs to be encrypted. This will also reduce

the overall size of the log file. The **encrypt-file** tool (or custom code, using the LDAP SDK's **com.unboundid.util.PassphraseEncryptedInputStream**) is used to access the encrypted data.

To enable encryption, at least one encryption settings definition must be defined in the server. Use the one created during setup, or create a new one with the **encryption-settings create** command. By default, the encryption will be performed with the server's preferred encryption settings definition. To explicitly specify which definition should be used for the encryption, the **encryption-settings-definition-id** property can be set with the ID of that definition. It is recommended that the encryption settings definition is created from a passphrase so that the file can be decrypted by providing that passphrase, even if the original encryption settings definition is no longer available. A randomly generated encryption settings definition can also be created, but the log file can only be decrypted using a server instance that has that encryption settings definition.

When using encrypted logging, a small amount of data may remain in an in-memory buffer until the log file is closed. The encryption is performed using a block cipher, and it cannot write an incomplete block of data until the file is closed. This is not an issue for any log file that is not being actively written. To examine the contents of a log file that is being actively written, use the **rotate-log** tool to force the file to be rotated before attempting to examine it.

Configuring log signing

Steps

1. Use **dsconfig** to enable log signing for a Log Publisher. In this example, set the `sign-log` property on the File-based Audit Log Publisher.

```
$ bin/dsconfig set-log-publisher-prop --publisher-name "File-Based Audit
 Logger" \
  --set sign-log:true
```

2. Disable and then re-enable the Log Publisher for the change to take effect.

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

Validating a signed file

About this task

The Directory Proxy Server provides a tool, **validate-file-signature**, that checks if a file has not been tampered with in any way.

Steps

- Run the **validate-file-signature** tool to check if a signed file has been tampered with. For this example, assume that the `sign-log` property was enabled for the File-Based Audit Log Publisher.

```
$ bin/validate-file-signature --file logs/audit
```

```
All signature information in file 'logs/audit' is valid
```

ⓘ **Note:** If any validations errors occur, you will see a message similar to the one as follows:

```
One or more signature validation errors were encountered
while validating the contents of file 'logs/audit':
* The end of the input stream was encountered without
```

```
  encountering the end of an active signature block.
  The contents of this signed block cannot be trusted
  because the signature cannot be verified
```

Configuring log file encryption

Steps

1. Use **dsconfig** to enable encryption for a Log Publisher. In this example, the File-based Access Log Publisher "Encrypted Access" is created, compression is set, and rotation and retention policies are set.

```
$ bin/dsconfig create-log-publisher-prop --publisher-name "Encrypted Access"
 \
  --type file-based-access \
  --set enabled:true \
  --set compression-mechanism:gzip \
  --set encryption-settings-definition-
id:332C846EF0DCD1D5187C1592E4C74CAD33FC1E5FC20B726CD301CDD2B3FFBC2B \
  --set encrypt-log:true \
  --set log-file:logs/encrypted-access \
  --set "rotation-policy:24 Hours Time Limit Rotation Policy" \
  --set "rotation-policy:Size Limit Rotation Policy" \
  --set "retention-policy:File Count Retention Policy" \
  --set "retention-policy:Free Disk Space Retention Policy" \
  --set "retention-policy:Size Limit Retention Policy"
```

2. To decrypt and decompress the file:

```
$ bin/encrypt-file --decrypt \
  --decompress-input \
  --input-file logs/encrypted-access.20180216040332Z.gz.encrypted \
  --output-file decrypted-access
Initializing the server's encryption framework...Done
Writing decrypted data to file '/ds/PingDirectoryProxy/decrypted-access'
 using a
key generated from encryption settings definition
 '332c846ef0dcd1d5187c1592e4c74cad33fc1e5fc20b726cd301cdd2b3ffbc2b'
Successfully wrote 123,456,789 bytes of decrypted data
```

Configuring log rotation

About this task

The Directory Proxy Server allows you to configure the log rotation policy for the server. When any rotation limit is reached, the Directory Proxy Server rotates the current log and starts a new log. If you create a new log publisher, you must configure at least one log rotation policy.

You can select the following properties:

- **Time Limit Rotation Policy**. Rotates the log based on the length of time since the last rotation. Default implementations are provided for rotation every 24 hours and every 7 days.
- **Fixed Time Rotation Policy**. Rotates the logs every day at a specified time (based on 24-hour time). The default time is 2359.
- **Size Limit Rotation Policy**. Rotates the logs when the file reaches the maximum size for each log. The default size limit is 100 MB.
- **Never Rotate Policy**. Used in a rare event that does not require log rotation.

Steps

- Use **dsconfig** to modify the log rotation policy for the access logger.

```
$ bin/dsconfig set-log-publisher-prop \
  --publisher-name "File-Based Access Logger" \
  --remove "rotation-policy:24 Hours Time Limit Rotation Policy" \
  --add "rotation-policy:7 Days Time Limit Rotation Policy"
```

Configuring log rotation listeners

The Directory Proxy Server provides two log file rotation listeners: the copy log file rotation listener and the summarize log file rotation listener, which can be enabled with a log publisher. Log file rotation listeners allow the server to perform a task on a log file as soon as it has been rotated out of service. Custom log file listeners can be created with the Server SDK.

The copy log file rotation listener can be used to compress and copy a recently-rotated log file to an alternate location for long-term storage. The original rotated log file will be subject to deletion by a log file retention policy, but the copy will not be automatically removed. Use the following command to create a new copy log file rotation listener:

```
$ dsconfig create-log-file-rotation-listener \
  --listener-name "Copy on Rotate" \
  --type copy \
  --set enabled:true \
  --set copy-to-directory:/path/to/archive/directory \
  --set compress-on-copy:true
```

The path specified by the **copy-to-directory** property must exist, and the file system containing that directory must have enough space to hold all of the log files that will be written there. The server automatically monitors free disk space on the target file system and generates administrative alerts if the amount of free space gets too low.

The summarize log file rotation listener invokes the **summarize-access-log** tool on a recently-rotated log file and writes its output to a file in a specified location. This provides information about the number and types of operations processed by the server, processing rates and response times, and other useful metrics. Use this with access loggers that log in a format that is compatible with the **summarize-access-log** tool, including the **file-based-access** and **operation-timing-access** logger types. Use the following command to create a new summarize log file rotation listener:

```
$ dsconfig create-log-file-rotation-listener \
  --listener-name "Summarize on Rotate" \
  --type summarize \
  --set enabled:true \
  --set output-directory:/path/to/summary/directory
```

The summary output files have the same name as the rotated log file, with an extension of **.summary**. If the **output-directory** property is specified, the summary files are written to that directory. If not specified, files are placed in the directory in which the log files are written.

As with the copy log file rotation listener, summary files are not automatically deleted. Although files are generally small in comparison to the log files themselves, make sure that enough space is available in the specified storage directory. The server automatically monitors free disk space on the file system to which the summary files are written.

Configuring log retention

About this task

The Directory Proxy Server allows you to configure the log retention policy for each log on the server. When any retention limit is reached, the Directory Proxy Server removes the oldest archived log prior to

creating a new log. Log retention is only effective if you have a log rotation policy in place. If you create a new log publisher, you must configure at least one log retention policy.

- **File Count Retention Policy**. Sets the number of log files you want the Directory Proxy Server to retain. The default file count is 10 logs. If the file count is set to 1, then the log will continue to grow indefinitely without being rotated.
- **Free Disk Space Retention Policy**. Sets the minimum amount of free disk space. The default free disk space is 500 MBytes.
- **Size Limit Retention Policy**. Sets the maximum size of the combined archived logs. The default size limit is 500 MBytes.
- **Time Limit Retention Policy**. Sets the maximum length of time that rotated log files should be retained.
- **Custom Retention Policy**. Create a new retention policy that meets your Directory Proxy Server's requirements. This will require developing custom code to implement the desired log retention policy.
- **Never Delete Retention Policy**. Used in a rare event that does not require log deletion.

Steps

- Use **`dsconfig`** to modify the log retention policy for the access logger.

```
$ bin/dsconfig set-log-publisher-prop \
  --publisher-name "File-Based Access Logger" \
  --set "retention-policy:Free Disk Space Retention Policy"
```

Setting resource limits

You can set resource limits for the Directory Proxy Server using several global configuration properties as well as setting resource limits on specific client connection policies. If you configure both global and client connection policy resource limits, the first limit reached will always be honored. For example, if the server-wide maximum concurrent connections limit is reached, then all subsequent connection will be rejected until existing connections are closed, regardless of whether a client connection policy limit has been reached.

Setting global resource limits

You can specify the following types of global resource limits:

- Specify the maximum number of client connections that can be established at any given time using the `maximum-concurrent-connections` property. If the server already has the maximum number of connections established, then any new connection attempts from any clients will be rejected until an existing connection is closed. The default value of zero indicates that no limit is enforced.
- Specify the maximum number of client connections that can be established at any give time from the same client system using the `maximum-concurrent-connections-per-ip-address` property. If the server already has the maximum number of connections established from a given client, then any new connection attempts from that client will be rejected until an existing connection from that client is closed. The server may continue to accept connections from other clients that have not yet reached this limit. The default value of zero indicates that no limit is enforced.
- Specify the maximum number of client connections that can be established at any given time while authenticated as a particular user with the `maximum-concurrent-connections-per-bind-dn` property. This property applies after the connection is established, because the bind operation to authenticate the user happens after the connection is established rather than during the course of establishing the connection itself. If the maximum number of connections are authenticated as a given user, then any new attempt to authenticate as that user will cause the connection performing the bind to be terminated. Note that this limit applies only to authenticated connections, and will not be enforced for clients that have not authenticated or for clients that have authenticated as the anonymous user. The default value of zero indicates that no limit is enforced.

Any changes to the `maximum-concurrent-connections` and `maximum-concurrent-connections-per-ip-address` properties will take effect only for new connections established after the change is made. Any change to the `maximum-concurrent-connections-per-bind-dn` property will apply only to connections (including existing connections) which perform authentication after the change is made. Existing connections will be allowed to remain established even if that would cause the new limit to be exceeded.

Setting client connection policy resource limits

You can also configure resource limits in a client connection policy using the following properties of the client connection policy:

- **maximum-concurrent-connections**. This property specifies the maximum number of client connections that may be associated with a specific client connection policy at any given time. Once this limit has been reached, any further attempts to associate a connection with this client connection policy will result in the termination of the connection.
- **maximum-connection-duration**. This property specifies the maximum length of time that a connection associated with a particular client connection policy may be established. When the connection has been established longer than this period, it will be terminated.
- **maximum-idle-connection-duration**. This property specifies the maximum time that a connection associated with a particular client connection policy may remain established after the completion of the last operation processed on that connection. Any new operation requested on the connection resets the timer. Connections that are idle for longer than the specified time will be terminated.
- **maximum-operation-count-per-connection**. This property specifies the maximum number of operations that may be requested by any client connection associated with this client connection policy. If an attempt is made to process more than this number of operations on the connection, then the connection will be terminated.
- **maximum-concurrent-operations-per-connection**. This property specifies the maximum number of concurrent operations that can be in progress for any connection. This property can be used to prevent a single client connection from monopolizing server processing resources by sending a large number of concurrent asynchronous requests.
- **maximum-connection-operation-rate**. This property specifies the maximum rate at which a client associated with a specific client connection policy may issue requests to the Directory Proxy Server. If a client attempts to request operations at a rate higher than this limit, then the server will behave as described by the `connection-operation-rate-exceeded-behavior` property.
- **connection-operation-rate-exceeded-behavior**. This property describes how the server should behave if a client connection attempts to exceed a rate defined in the `maximum-connection-operation-rate` property.
- **maximum-policy-operation-rate**. This property specifies the maximum rate at which all clients associated with a particular client connection policy may issue requests to the Directory Proxy Server. If this limit is exceeded, then the server will exhibit the behavior described in the `policy-operation-rate-exceeded-behavior` property.
- **policy-operation-rate-exceeded-behavior**. This property specifies the behavior of the Directory Proxy Server if a client connection attempts to exceed the rate defined in the `maximum-policy-operation-rate` property.

Monitoring the Directory Proxy Server

While the Directory Proxy Server is running, it generates a significant amount of information available through monitor entries. This section contains information about the following:

- Monitoring Server Status Using the status Tool
- About the Monitor Entries
- Using the Monitoring Interfaces
- Monitoring with JMX

Monitoring system data using the PingDataMetrics Server

The PingDataMetrics Server provides collection and storage of performance data from your server topology. You can use the System Utilization Monitor with the PingDataMetrics Server to collect information about the host system CPU, disk, and network utilization on any platform except Linux. If you are not using the PingDataMetrics Server, you do not need to use the system utilization monitor. When data is being collected, it periodically forks the process and executes commands.

For more information about using the System Utilization Monitor, refer to the data collection chapter of the PingDataMetrics Server documentation.

Monitoring the server using the status tool

About this task

The PingDirectoryProxy Server provides a **`status`** tool that provides basic server status information, including version, connection handlers, a table of LDAP external servers, and the percent of the global index that is used.

Steps

1. Run the **`status`** tool to view the current state of the server.

```
$ bin/status
```

2. Enter the LDAP connection parameters.

```
>>>> Specify LDAP connection parameters

Administrator user bind DN [cn=Directory Manager]:

Password for user 'cn=Directory Manager':
```

```
          --- Server Status ---
Server Run Status:    Started 07/Jan/2011:10:59:52.000 -0600
Operational Status:   Available
Open Connections:     4
Max Connections:      8
Total Connections:    25

          --- Server Details ---
Host Name:            example
Administrative Users: cn=Directory Manager
Installation Path:    /path/to/PingDirectoryProxy
Version:              PingDirectoryProxy Server
                  8.0.0.0
Java Version:         jdk-7u9

          --- Connection Handlers ---

Address:Port : Protocol : State
-------------:----------:---------

0.0.0.0:1689 : JMX      : Disabled
0.0.0.0:636  : LDAPS    : Disabled
0.0.0.0:9389 : LDAP     : Enabled

         --- LDAP External Servers ---

Server         : Status    : Score : LB Algorithm
---------------:-----------:-------:---------------------
localhost:389  : Available : 10    : dc_example_dc_com-failover
localhost:1389 : Available : 10    : dc_example_dc_com-failover
```

```
        --- LDAP External Server Op Counts ---

Server         : Add : Bind:Compare:Delete:Modify:Mod DN:Search : All
---------------:-----:----:------:-------:-------:------:-------:----
localhost:11389: 0   : 0  : 0    : 0     : 0     : 0    : 1249  : 1249
localhost:12389: 0   : 0  : 0    : 0     : 0     : 0    : 494   : 494



        --- Entry Balancing Request Processors ---

 Base DN                      : Global Index % Used
------------------------------:--------------------
ou=people,dc=example,dc=com : 33


        --- Global Index Stats for ou=people,dc=example,dc=com ---

Index : Total Bytes : Key Bytes : Keys : Size (# Keys) : Inserted :
Removed : Replaced: Hits : Misses : Discarded : Duplicates
------:-------------:-----------:---------:---------------:----------
rdn   : 30667304    : 14888906  : 1000001 : 3464494 0 :0 :0 :0 :0 :0
uid   : 26523480    : 10888902  : 1000001 : 3464494 0 : 0 : 3583 : 0 : 0 : 0


         --- Operation Processing Time ---

Op Type   : Total Ops : Avg Resp Time (ms)
----------:-----------:-------------------
Add       : 0         : 0.0
Bind      : 0         : 0.0
Compare   : 0         : 0.0
Delete    : 0         : 0.0
Modify    : 0         : 0.0
Modify DN : 0         : 0.0
Search    : 3583      : 117.58
All       : 3583      : 117.58


         --- Work Queue ---

          : Recent : Average : Maximum
----------:--------:---------:--------
Queue Size : 0      : 0       : 1
% Busy     : 0      : 1       : 19
```

About the monitor entries

While the Directory Proxy Server is running, it generates a significant amount of information available through monitor entries. Monitor entries are available over LDAP in the `cn=monitor` subtree. The types of monitor entries that are available include:

- **General Monitor Entry (cn=monitor)** – Provides a basic set of general information about the server.
- **Active Operations Monitor Entry (cn=Active Operations,cn=monitor)** – Provides information about all operations currently in progress in the server.
- **Backend Monitor Entries (cn={id} Backend,cn=monitor)** – Provides information about the backend, including the number of entries, the base DN(s), and whether it is private.
- **Client Connections Monitor Entry (cn=Client Connections,cn=monitor)** – Provides information about all connections currently established to the server.

- **Connection Handler Monitor Entry (cn={name},cn=monitor)** – Provides information about the configuration of each connection handler and the client connections established to it.
- **Database Environment Monitor Entries (cn={id} Database Environment,cn=monitor)** – Provides statistics and other data from the Oracle Berkeley DB Java Edition database envi- ronment used by the associated backend.
- **Disk Space Usage Monitor Entry (cn=Disk Space Usage,cn=monitor)** – Provides infor- mation about the amount of usable disk space available to server components.
- **JVM Memory Usage Monitor Entry (cn=JVM Memory Usage,cn=monitor)** – Provides information about garbage collection activity, the amount of memory available to the server, and the amount of memory consumed by various server components.
- **JVM Stack Trace Monitor Entry (cn=JVM Stack Trace,cn=monitor)** – Provides a stack trace of all threads in the JVM.
- **LDAP Statistics Monitor Entries (cn={name} Statistics,cn=monitor)** – Provides information about the number of each type of operation requested and bytes transferred over the connection handler.
- **Processing Time Histogram Monitor Entry (cn=Processing Time Histogram,cn=monitor)** – Provides information about the number of percent of operations that completed in various response time categories.
- **SSL Context Monitor Entry (cn=SSL Context,cn=monitor)** – Provides information about the available and supported SSL Cipher Suites and Protocols on the server.
- **System Information Monitor Entry (cn=System Information,cn=monitor)** – Provides information about the underlying JVM and system.
- **Version Monitor Entry (cn=Version,cn=monitor)** – Provides information about the Directory Proxy Server version.
- **Work Queue Monitor Entry (cn=Work Queue,cn=monitor)** – Provides information about the state of the Directory Proxy Server work queue, including the number of operations waiting on worker threads and the number of operations that have been rejected because the queue became full.

Working with alarms, alerts, and gauges

An alarm represents a stateful condition of the server or a resource that may indicate a problem, such as low disk space or external server unavailability. A gauge defines a set of threshold values with a specified severity that, when crossed, cause the server to enter or exit an alarm state. Gauges are used for monitoring continuous values like CPU load or free disk space (Numeric Gauge), or an enumerated set of values such as 'server unavailable' or ‘server unavailable’ (Indicator Gauge). Gauges generate alarms, when the gauge’s severity changes due to changes in the monitored value. Like alerts, alarms have severity (NORMAL, WARNING, MINOR, MAJOR, CRITICAL), name, and message. Alarms will always have a Condition property, and may have a Specific Problem or Resource property. If surfaced through SNMP, a Probable Cause property and Alarm Type property are also listed. Alarms can be configured to generate alerts when the alarm's severity changes. The Alarm Manager, which governs the actions performed when an alarm state is entered, is configurable through the **`dsconfig`** tool and Administrative Console. A complete listing of system alerts, alarms, and their severity is available in `<server-root>/docs/admin-alerts-list.csv`.

There are two alert types supported by the server - standard and alarm-specific. The server constantly monitors for conditions that may need attention by administrators, such as low disk space. For this condition, the standard alert is **`low-disk-space-warning`**, and the alarm-specific alert is **`alarm-warning`**. The server can be configured to generate alarm-specific alerts instead of, or in addition to, standard alerts. By default, standard alerts are generated for conditions internally monitored by the server. However, gauges can only generate alarm-alerts.

The Directory Proxy Server installs a set of gauges that are specific to the product and that can be cloned or configured through the **`dsconfig`** tool. Existing gauges can be tailored to fit each environment by adjusting the update interval and threshold values. Configuration of system gauges determines the criteria by which alarms are triggered. The Stats Logger can be used to view historical information about the value and severity of all system gauges.

The Directory Proxy Server is compliant with the International Telecommunication Union CCITT Recommendation X.733 (1992) standard for generating and clearing alarms. If configured, entering or exiting an alarm state can result in one or more alerts. An alarm state is exited when the condition no longer applies. An `alarm_cleared` alert type is generated by the system when an alarm's severity changes from a non-normal severity to any other severity. An `alarm_cleared` alert will correlate to a previous alarm when the Condition and Resource properties are the same. The Condition corresponds to the Summary column in the `admin-alerts-list.csv` file.

Like the Alerts Backend, which stores information in **`cn=alerts`**, the Alarm Backend stores information within the **`cn=alarms`** backend. Unlike alerts, alarm thresholds have a state over time that can change in severity and be cleared when a monitored value returns to normal. Alarms can be viewed with the **`status`** tool. As with other alert types, alert handlers can be configured to manage the alerts generated by alarms.

Testing alarms and alerts

Steps

1. Configure a gauge with **`dsconfig`** and set the **`override-severity`** property to critical. The following example uses the CPU Usage (Percent) gauge.

```
$ dsconfig set-gauge-prop \
  --gauge-name "CPU Usage (Percent)" \
  --set override-severity:critical
```

2. Run the **`status`** tool to verify that an alarm was generated with corresponding alerts. The **`status`** tool provides a summary of the server's current state with key metrics and a list of recent alerts and alarms. The sample output has been shortened to show just the alarms and alerts information.

```
$ bin/status

                      --- Administrative Alerts ---
Severity : Time             : Message
---------:------------------:-----------------------------------------------------
Info     : 11/Aug/2014      : A configuration change has been made in the
         : 15:48:46 -0500   : Directory Server:
         :                  : [11/Aug/2014:15:48:46.054 -0500]
         :                  : conn=17 op=73 dn='cn=Directory Manager,cn=Root
         :                  : DNs,cn=config' authtype=[Simple]
from=127.0.0.1
         :                  : to=127.0.0.1 command='dsconfig set-gauge-prop
         :                  :  --gauge-name 'Cleaner Backlog (Number Of
Files)'
         :                  : --set warning-value:-1'
Info     : 11/Aug/2014      : A configuration change has been made in the
         :  15:47:32 -0500  : Directory Server: [11/Aug/2014:15:47:32.547
-0500]
         :                  : conn=4 op=196 dn='cn=Directory Manager,cn=Root
         :                  : DNs,cn=config' authtype=[Simple]
from=127.0.0.1
         :                  : to=127.0.0.1 command='dsconfig set-gauge-prop
         :                  : --gauge-name 'Cleaner Backlog (Number Of
Files)'
         :                  :  --set warning-value:0'
Error    : 11/Aug/2014      : Alarm [CPU Usage (Percent). Gauge CPU Usage
(Percent)
         :  15:41:00 -0500  : for Host System has
         :                  : a current value of '18.583333333333332'.
         :                  : The severity is currently OVERRIDDEN in the
         :                  : Gauge's configuration to 'CRITICAL'.
         :                  : The actual severity is: The severity is
         :                  : currently 'NORMAL', having assumed this
severity
```

```
         :                  : Mon Aug 11 15:41:00 CDT 2014. If CPU use is
high,
         :                  : check the server's current workload and make
any
         :                  : needed adjustments. Reducing the load on the
system
         :                  : will lead to better response times.
         :                  : Resource='Host System']
         :                  : raised with critical severity
Shown are alerts of severity [Info,Warning,Error,Fatal] from the past 48
 hours
Use the --maxAlerts and/or --alertSeverity options to filter this list
```

```
                         --- Alarms ---
Severity : Severity Start : Condition : Resource    : Details
         : Time           :           :              :
---------:----------------:-----------:--------------:----------------------------
Critical : 11/Aug/2014    : CPU Usage : Host System : Gauge CPU Usage
(Percent) for
         : 15:41:00 -0500 : (Percent) :              : Host System
         :                :           :              : has a current value
of
         :                :           :              : '18.785714285714285'.
         :                :           :              : The severity is
currently
         :                :           :              : 'CRITICAL', having
assumed
         :                :           :              : this severity Mon Aug
11
         :                :           :              : 15:49:00 CDT 2014. If
CPU use
         :                :           :              : is high, check the
server's
         :                :           :              : current workload and
make any
         :                :           :              : needed adjustments.
Reducing
         :                :           :              : the load on the
system will
         :                :           :              : lead to better
response times
Warning  : 11/Aug/2014    : Work Queue: Work Queue   : Gauge Work Queue Size
(Number
         : 15:39:40 -0500 : Size      :              : of Requests) for Work
Queue
         :                : (Number of:              : has a current value
of '27'.
         :                : Requests) :              : The severity is
currently
         :                :           :              : 'WARNING' having
assumed this
         :                :           :              : severity Mon Aug 11
15:48:50
         :                :           :              : CDT 2014. If all
worker
         :                :           :              : threads are busy
processing
         :                :           :              : other client
requests, then
         :                :           :              : new requests that
arrive will
         :                :           :              : be forced to wait in
the work
```

```
          :                 :           :              : queue until a worker
 thread
          :                 :           :              : becomes available
Shown are alarms of severity [Warning,Minor,Major,Critical]
Use the --alarmSeverity option to filter this list
```

Indeterminate alarms

Indeterminate alarms are raised for a server condition for which a severity cannot be determined. In most cases these alarms are benign and do not issue alerts nor appear in the output of the **`status`** tool or Administrative Console by default. These alarms are usually caused by an enabled gauge that is intended to measure an aspect of the server that is not currently enabled. For example, gauges intended to monitor metrics related to replication may produce indeterminate alarms if a Directory Server is not currently replicating data. The gauge can be disabled if needed.

For more information about indeterminate alarms, view the gauge's associated monitor entry. There may be messages that can help determine the issue. The following is sample output from the **`status`** tool run with the **`—alarmSeverity=indeterminate`** option:

```
                         --- Alarms ---
Severity     : Severity Start : Condition      : Resource   : Details
             : Time           :                :            :
-------------:----------------:----------------:------------:-----------------------
Normal       : 26/Aug/2014    : Startup Begun  : cn=config  : The Directory
 Server
             : 14:16:29 -0500 :                :            : is starting.
             :                :                :            :
Indeterminate: 26/Aug/2014    : Replication    : not        : The value of
 gauge
             : 14:16:40 -0500 : Latency        : available  : Replication
 Latency
             :                : (Milliseconds) :            : (Milliseconds)
 could not
             :                :                :            : be determined.
 The
             :                :                :            : severity is
 INDETERMINATE,
             :                :                :            : having assumed
 this
             :                :                :            : severity Tue Aug
 26
             :                :                :            : 14:17:10 CDT
 2014.
```

The following is an indeterminate alarm for the Replication Latency (Milliseconds) gauge. The following is a sample search of the monitor backend for this gauge’s entry. The result is an error message may explain the indeterminate severity:

```
# ldapsearch -w password --baseDN "cn=monitor"  \
-D"cn=directory manager" gauge-name="Replication Latency (Milliseconds)"

dn: cn=Gauge Replication Latency (Milliseconds),cn=monitor
objectClass: top
objectClass: ds-monitor-entry
objectClass: ds-numeric-gauge-monitor-entry
objectClass: ds-gauge-monitor-entry
objectClass: extensibleObject
cn:          Gauge Replication Latency (Milliseconds)
gauge-name:  Replication Latency (Milliseconds)
resource:
severity:    indeterminate
summary:     The value of gauge Replication Latency (Milliseconds) could not
             be determined. The severity is INDETERMINATE, having assumed
```

```
             this severity Tue Aug 26 15:42:40 CDT 2014
error-message: No entries were found under cn=monitor having object
               class ds-replica-monitor-entry
              …
```

Working with Administrative Alert Handlers

The PingDirectoryProxy Server provides mechanisms to send alert notifications to administrators when significant problems or events occur during processing, such as problems during server startup or shutdown. The Directory Proxy Server provides a number of alert handler implementations, including:

- **Error Log Alert Handler**. Sends administrative alerts to the configured server error logger(s).
- **Exec Alert Handler**. Executes a specified command on the local system if an administrative alert matching the criteria for this alert handler is generated by the Directory Proxy Server. Information about the administrative alert will be made available to the executed application as arguments provided by the command.
- **Groovy Scripted Alert Handler**. Provides alert handler implementations defined in a dynamically-loaded Groovy script that implements the `ScriptedAlertHandler` class defined in the Server SDK.
- **JMX Alert Handler**. Sends administrative alerts to clients using the Java Management Extensions (JMX) protocol. Ping Identity uses JMX for monitoring entries and requires that the JMX connection handler be enabled.
- **SMTP Alert Handler**. Sends administrative alerts to clients via email using the Simple Mail Transfer Protocol (SMTP). The server requires that one or more SMTP servers be defined in the global configuration.
- **SNMP Alert Handler**. Sends administrative alerts to clients using the Simple Network Monitoring Protocol (SNMP). The server must have an SNMP agent capable of communicating via SNMP 2c.
- **SNMP Subagent Alert Handler**. Sends SNMP traps to a master agent in response to administrative alerts generated within the server.
- **Third Party Alert Handler**. Provides alert handler implementations created in third-party code using the Server SDK.

Configuring the JMX Connection Handler and Alert Handler

You can configure the JMX connection handler and alert handler respectively using the **`dsconfig`** tool. Any user allowed to receive JMX notifications must have the `jmx-read` and `jmx-notify` privileges. By default, these privileges are not granted to any users (including root users or global administrators). For security reasons, we recommend that you create a separate user account that does not have any other privileges but these. Although not shown in this section, you can configure the JMX connection handler and alert handler using **`dsconfig`** in interactive command-line mode, which is visible on the "Standard" object menu.

Configuring the JMX Connection Handler

Steps

1. Use **`dsconfig`** to enable the JMX Connection Handler.

```
$ bin/dsconfig set-connection-handler-prop \
  --handler-name "JMX Connection Handler" \
  --set enabled:true \
  --set listen-port:1689
```

2. Add a new non-root user account with the `jmx-read` and `jmx-notify` privileges. This account can be added using the **`ldapmodify`** tool using an LDIF representation like:

```
dn: cn=JMX User,cn=Root DNs,cn=config
changetype: add
objectClass: top
objectClass: person
objectClass: organizationalPerson
```

```
objectClass: inetOrgPerson
objectClass: ds-cfg-root-dn-user
givenName: JMX
sn: User
cn: JMX User
userPassword: password
ds-cfg-inherit-default-root-privileges: false
ds-cfg-alternate-bind-dn: cn=JMX User
ds-privilege-name: jmx-read
ds-privilege-name: jmx-notify
```

Configuring the JMX Alert Handler

Steps

- Use **dsconfig** to configure the JMX Alert Handler.

```
$ bin/dsconfig set-alert-handler-prop --handler-name "JMX Alert Handler" \
  --set enabled:true
```

Configuring the SMTP Alert Handler

By default, there is no configuration entry for an SMTP alert handler. To create a new instance of an SMTP alert handler, use the **dsconfig** tool.

Configuring the SMTP Alert Handler

Steps

- Use the **dsconfig** tool to configure the SMTP Alert Handler.

```
$ bin/dsconfig create-alert-handler \
  --handler-name "SMTP Alert Handler" \
  --type smtp \
  --set enabled:true \
  --set "sender-address:alerts@example.com" \
  --set "recipient-address:administrators@example.com" \
  --set "message-subject:Directory Admin Alert \%\%alert-type\%\%" \
  --set "message-body:Administrative alert:\\n\%\%alert-message\%\%"
```

Configuring the SNMP Subagent Alert Handler

You can configure the SNMP Subagent alert handler using the **dsconfig** tool, which is visible at the "Standard" object menu. Before you begin, you need an SNMP Subagent capable of communicating via SNMP2c. For more information on SNMP, see Monitoring Using SNMP.

To Configure the SNMP Subagent Alert Handler

Steps

- Use **dsconfig** to configure the SNMP subagent alert handler. The `server-host-name` is the address of the system running the SNMP subagent. The `server-port` is the port number on which the subagent is running. The `community-name` is the name of the SNMP community that is used for the traps.

  The Directory Proxy Server also supports a SNMP Alert Handler, which is used in deployments that do not enable an SNMP subagent.

```
$ bin/dsconfig set-alert-handler-prop \
  --handler-name "SNMP Subagent Alert Handler" \
  --set enabled:true \
  --set server-host-name:host2 \
  --set server-port:162 \
```

```
  --set community-name:public
```

Working with virtual attributes

The PingDirectoryProxy Server provides dynamically generated attributes called virtual attributes for local Directory Proxy Server data. The proxy virtual attributes apply to a local proxy backend, such as `cn=config` or the Root DSE. If you want to have virtual attributes in entries for proxied requests, then they must be configured in the backend servers. Alternately, attributes may be inserted into those entries using proxy transformations. For more information about configuring proxy transformations, see "Configuring Proxy Transformations".

For example, you can define a virtual attribute and assign it to the Root DSE as follows:

```
$ bin/dsconfig create-virtual-attribute \
  --name defineDescriptionOnRootDSE --type user-defined \
  --set enabled:true --set attribute-type:description \
  --set filter:objectclass=ds-root-dse --set value:PrimaryProxy
```

If you search the Root DSE using the following LDAP search, you see that the description attribute now has the value `PrimaryProxy`.

```
$ bin/ldapsearch --baseDN "" --searchScope base --bindDN "" \
  --bindPassword "" --port 5389 -- hostname localhost \
  "objectclass=*" description

dn:
description:PrimaryProxy
```

Managing Monitoring

The PingDirectoryProxy Server also provides a flexible monitoring framework that exposes its monitoring information under the `cn=monitor` entry and provides interfaces via the PingDataMetrics™ Server, the Administrative Console, SNMP, JMX, and over LDAP. The Directory Proxy Server also provides a tool, the Periodic Stats Logger, to profile server performance.

This chapter presents the following information:

The monitor backend

The Directory Proxy Server exposes its monitoring information under the `cn=monitor` entry. Administrators can use various means to monitor the servers, including the PingDataMetrics Server, through SNMP, the Administrative Console, JConsole, LDAP command-line tools, and the Periodic Stats Logger. Use the **`bin/status`** tool to display server component activity and state.

The list of all monitor entries can be seen using ldapsearch as follows:

```
$ bin/ldapsearch --hostname server1.example.com --port 1389 \
 --bindDN "uid=admin,dc=example,dc=com" --bindPassword secret \
 --baseDN "cn=monitor" "(objectclass=*)" cn
```

The following table describes a subset of the monitor entries:

Directory Proxy Server Monitoring Components

Component	Description
Active Operations	Provides information about the operations currently being processed by the Directory Proxy Server. Shows the number of operations, information on each operation, and the number of active persistent searches.

Component	Description
Backends	Provides general information about the state of an a Directory Proxy Server backend, including the entry count. If the backend is a local database, there is a corresponding database environment monitor entry with information on cache usage and on-disk size.
Client Connections	Provides information about all client connections to the Directory Proxy Server. The client connection information contains a name followed by an equal sign and a quoted value (e.g., connID="15", connectTime="20100308223038Z", etc.)
Connection Handlers	Provides information about the available connection handlers on the Directory Proxy Server, which includes the LDAP and LDIF connection handlers. These handlers are used to accept client connections and to read requests and send responses to those clients.
Disk Space Usage	Provides information about the disk space available to various components of the Directory Proxy Server.
General	Provides general information about the state of the Directory Proxy Server, including product name, vendor name, server version, etc.
Index	Provides on each index. The monitor captures the number of keys preloaded, and counters for read/write/remove/open-cursor/read-for-search. These counters provide insight into how useful an index is for a given workload.
HTTP/HTTPS Connection Handler Statistics	Provides statistics about the interaction that the associated HTTP connection handler has had with its clients, including the number of connections accepted, average requests per connection, average connection duration, total bytes returned, and average processing time by status code.
JVM Stack Trace	Provides a stack trace of all threads processing within the JVM.
LDAP Connection Handler Statistics	Provides statistics about the interaction that the associated LDAP connection handler has had with its clients, including the number of connections established and closed, bytes read and written, LDAP messages read and written, operations initiated, completed, and abandoned, etc.
Processing Time Histogram	Categorizes operation processing times into a number of user-defined buckets of information, including the total number of operations processed, overall average response time (ms), number of processing times between 0ms and 1ms, etc.
System Information	Provides general information about the system and the JVM on which the Directory Proxy Server is running, including system host name, operation system, JVM architecture, Java home, Java version, etc.
Version	Provides information about the Directory Proxy Server version, including build ID, version, revision number, etc.

Component	Description
Work Queue	Provides information about the state of the Directory Proxy Server work queue, which holds requests until they can be processed by a worker thread, including the requests rejected, current work queue size, number of worker threads, and number of busy worker threads. The work queue configuration has a **`monitor-queue-time`** property set to **`true`** by default. This logs messages for new operations with a **`qtime`** attribute included in the log messages. Its value is expressed in milliseconds and represents the length of time that operations are held in the work queue. A dedicated thread pool can be used for processing administrative operations. This thread pool enables diagnosis and corrective action if all other worker threads are processing operations. To request that operations use the administrative thread pool, using the **`ldapsearch`** command for example, use the **`--useAdministrativeSession`** option. The requester must have the **`use-admin-session`** privilege (included for root users). By default, eight threads are available for this purpose. This can be changed with the **`num-administrative-session-worker-threads`** property in the work queue configuration.

Monitoring disk space usage

The disk space usage monitor provides information about the amount of usable disk space available for Directory Proxy Server components. The disk space usage monitor evaluates the free space at locations registered through the `DiskSpaceConsumer` interface by various components of the server. Disk space monitoring excludes disk locations that do not have server components registered. However, other disk locations may still impact server performance, such as the operating system disk, if it becomes full. When relevant to the server, these locations include the server root, the location of the `/config` directory, the location of every log file, all JE backend directories, the location of the changelog, the location of the replication environment database, and the location of any server extension that registers itself with the DiskSpaceConsumer interface.

The disk space usage monitor provides the ability to generate administrative alerts, as well as take additional action if the amount of usable space drops below the defined thresholds.

Three thresholds can be configured for this monitor:

- **Low space warning threshold**. This threshold is defined as either a percentage or absolute amount of usable space. If the amount of usable space drops below this threshold, then the Directory Proxy Server will generate an administrative alert but will remain fully functional. It will generate alerts at regular intervals that you configure (such as once a day) unless action is taken to increase the amount of usable space. The Directory Proxy Server will also generate additional alerts as the amount of usable space is further reduced (e.g., each time the amount of usable space drops below a value 10% closer to the low space error threshold). If an administrator frees up disk space or adds additional capacity, then the server should automatically recognize this and stop generating alerts.
- **Low space error threshold**. This threshold is also defined as either a percentage or absolute size. Once the amount of usable space drops below this threshold, then the server will generate an alert notification and will begin rejecting all operations requested by non-root users with "UNAVAILABLE" results. The server should continue to generate alerts during this time. Once the server enters this mode, then an administrator will have to take some kind of action (e.g., running a command to invoke a task or removing a signal file) before the server will resume normal operation. This threshold must be less than or equal to the low space warning threshold. If they are equal, the server will begin rejecting requests from non-root users immediately upon detecting low usable disk space.
- **Out of space error threshold**. This threshold may also be defined as a percentage or absolute size. Once the amount of usable space drops below this threshold, then the PingDirectoryProxy Server will generate a final administrative alert and will shut itself down. This threshold must be less than or equal

to the low space error threshold. If they are equal, the server will shut itself down rather than rejecting requests from non-root users.

- **Disk space monitoring for tools**. The server monitors disk space consumption during processing for the `export-ldif`, `rebuild-index`, and `backup` tools. Space is monitored every 10 seconds if usable space for all monitored paths is greater than 15 percent of the capacity of those volumes. If usable space for any path drops below 15 percent, or below 10GB free, the space check frequency is increased to every second. Warning messages are generated if available space falls below 10 percent, or below 5GB free. If usable space for any path drops below two percent, or 1GB free, the tool processing is aborted and files may be removed to free up space.

The default configuration uses the same values for the low space error threshold and out of space error threshold. This is to prevent having the server online but rejecting requests, which will cause problems with applications trying to interact with the server. The low space warning threshold generates an alert before the problem becomes serious, well in advance of available disk space dropping to a point that it is critical.

The default values may not be suitable for all disk sizes, and should be adjusted to fit the deployment. Determining the best values should factor in the size of the disk, how big the database may become, how much space log files may consume, and how many backups will be stored.

The threshold values may be specified either as absolute sizes or as percentages of the total available disk space. All values must be specified as absolute values or as percentages. A mix of absolute values and percentages cannot be used. The low space warning threshold must be greater than or equal to the low space error threshold, the low space error threshold must be greater than or equal to the out of space error threshold, and the out of space error threshold must be greater than or equal to zero.

If the out of space error threshold is set to zero, then the server will not attempt to automatically shut itself down if it detects that usable disk space has become critically low. If the amount of usable space reaches zero, then the database will preserve its integrity but may enter a state in which it rejects all operations with an error and requires the server (or at least the affected backends) to be restarted. If the low space error threshold is also set to zero, then the server will generate periodic warnings about low available disk space but will remain fully functional for as long as possible. If all three threshold values are set to zero, then the server will not attempt to warn about or otherwise react to a lack of usable disk space.

Monitoring with the PingDataMetrics Server

The PingDataMetrics Server is an invaluable tool for collecting, aggregating and exposing historical and instantaneous data from the various Ping Identity servers in a deployment. The PingDataMetrics Server relies on a captive PostgreSQL datastore for the metrics, which it collects from internal instrumentation across the instances, replicas, and data centers in your environment. The data is available via a Monitoring API that can be used to build custom dashboards and monitoring applications to monitor the overall health of your Ping Identity Platform system. For more information, refer to the *PingDataMetrics Server Administration Guide*.

Monitoring key performance indicators by application

The PingDirectoryProxy Server can be configured to track many key performance metrics (for example, throughput and response-time) by the client applications requesting them. This feature is invaluable for measuring whether the Ping Identity identify infrastructure meets all of your service-level agreements (SLA) that have been defined for client applications.

When enabled, the per-application monitoring data can be accessed in the `cn=monitor` backend, the Periodic Stats Logger, and made available for collection by the Metrics Server. See the “Profiling Server Performance Using the Periodic Stats Logger” for more information on using that component. Also, see the Directory Proxy Server Configuration section of the *PingDataMetrics Server Administration Guide* for details on configuring the server to expose metrics that interest you. Tracked application information is exposed in the PingDataMetrics Server by metrics having the 'application-name' dimension. See the documentation under `docs/metrics` of the PingDataMetrics Server for information on which metrics are available with the 'application-name' dimension.

Configuring the external Servers

Before you install the PingDataMetrics Server, you need to configure the servers you will be monitoring: PingDirectory Server, PingDirectoryProxy Server, and PingDataSync Server. The PingDataMetrics Server requires all servers to be version 3.5.0 or later. See the administration guides for each product for installation instructions.

Once you have installed the Directory Proxy Server, you can use the **dsconfig** tool to make configuration changes for the PingDataMetrics Server. When using the **dsconfig** tool interactively, set the complexity level to Advanced, so that you can make all the necessary configuration changes.

Preparing the servers monitored by the PingDataMetrics Server

The Metrics Backend manages the storage of metrics and provides access to the stored blocks of metrics via LDAP. The Metrics Backend is configured to keep a maximum amount of metric history based on log retention policies. The default retention policy uses the Default Size Limit Retention Policy, Free Disk Space Retention Policy, and the File Growth Limit Policy, limiting the total disk space used to 500 MB. This amount of disk typically contains more than 24 hours of metric history, which is ample. The Directory Proxy Server keeps a metric history so that the PingDataMetrics Server can be down for a period and then catch up when it comes back online.

The following two commands create a Retention Policy that limits the number of files to 2000, and sets the Metrics Backend to flush data to a new file every 30 seconds.

```
$ bin/dsconfig create-log-retention-policy \
  --policy-name StatsCollectorRetentionPolicy \
  --type file-count --set number-of-files:2000

$ bin/dsconfig set-backend-prop \
  --backend-name metrics --set sample-flush-interval:30s \
  --set retention-policy:StatsCollectorRetentionPolicy
```

These commands configure the Metrics Backend to keep 16 hours of metric history, which consumes about 250 MB of disk, ensuring that captured metrics are available to the PingDataMetrics Server within 30 seconds of when the metric was captured. The value of the `sample-flush-interval` attribute determines the maximum delay between when a metric is captured and when it can be picked up by the PingDataMetrics Server.

The flush interval can be set between 15 seconds and 60 seconds, with longer values resulting in less processing load on the PingDataMetrics Server. However, this flush interval increases the latency between when the metric was captured and when it becomes visible in the Dashboard Application. If you change the `sample-flush-interval` attribute to 60 seconds in the example above, then the Directory Proxy Server keeps 2000 minutes of history. Because the number of metrics produced per unit of time can vary depending on the configuration, no exact formula can be used to compute how much storage is required for each hour of history. However, 20 MB per hour is a good estimate.

Configuring the Processing Time Histogram plugin

The Processing Time Histogram plugin is configured on each Directory Proxy Server and Directory Proxy Server as a set of histogram bucket ranges. When the bucket ranges for a histogram change, the PingDataMetrics Server notices the change and marks samples differently. This process allows for histograms with the same set of bucket definitions to be properly aggregated and understood when returned in a query. If different servers have different bucket definitions, then a single metric query cannot return histogram data from the servers.

You should try to keep the Processing Time Histogram bucket definitions the same on all servers. Having different definitions restricts the ability of the PingDataMetrics Server API to aggregate histogram data across servers and makes the results of a query asking "What percentage of the search requests took less than 12 milliseconds?" harder to understand.

For each server in your topology, you must set the separate-monitor-entry-per-tracked-application property of the processing time histogram plugin to true. This property must be set to expose per-application

monitoring information under `cn=monitor`. When the separate-monitor-entry-per-tracked-application property is set to true, then the per-application-ldap-stats property must be set to per-application-only on the Stats Collector Plugin and vice versa.

For example, the following **dsconfig** command line sets the required properties of the Processing Time Histogram plugin:

```
$ bin/dsconfig set-plugin-prop --plugin-name "Processing Time Histogram" \
  --set separate-monitor-entry-per-tracked-application:true
```

The following **dsconfig** command line sets the per-application-ldap-stats property of the Stats Collector plugin to per-application-only:

```
$ bin/dsconfig set-plugin-prop --plugin-name "Stats Collector" \
 --set per-application-ldap-stats:per-application-only
```

Setting the connection criteria to collect SLA statistics by application

If you want to collect data about your SLAs, you need to configure connection criteria for each Service Level Agreement that you want to track. The connection criteria are used in many areas within the server. They are used by the client connection policies, but they can also be used when the server needs to perform matching based on connection-level properties, such as filtered logging. For assistance using connection criteria, contact your authorized support provider.

For example, imagine that we are interested in collecting statistics on data that is accessed by clients authenticating as the Directory Manager. We need to create connection criteria on the Directory Proxy Server that identifies any user authenticating as the Directory Manager. The connection criteria name corresponds to the application-name dimension value that clients will specify when accessing the data via the API. When you define the Connection Criteria, change the included-user-base-dn property to include the Directory Manager's full LDIF entry.

The following **dsconfig** command line creates connection criteria for the Directory Manager:

```
$ bin/dsconfig create-connection-criteria \
  --criteria-name "Directory Manager" \
  --type simple \
  --set "included-user-base-dn:cn=Directory Manager,cn=Root DNs,cn=config"
```

Updating the Global Configuration

You also need to create Global Configuration-tracked applications for each app (connection criteria) you intend to track. The `tracked-application` property allows individual applications to be identified in the server by connection criteria. The name of the tracked application is the same as the name you defined for the connection criteria.

For example, the following **dsconfig** command line adds the connection criteria we created in the previous step to the list of tracked applications:

```
$ bin/dsconfig set-global-configuration-prop \
  --set "tracked-application:Directory Manager"
```

The value of the `tracked-application` field corresponds to the value of the `application-name` dimension value that clients will specify when accessing the data via the API.

Proxy considerations for tracked applications

In a proxy environment, the criteria should be defined in the Directory Proxy Server since the Directory Proxy Server passes the application name through to the Directory Server in the intermediate client control. If a client of the Directory Proxy Server or Directory Server happens to use the intermediate client control, then the client name specified in the control will be used as the application name regardless of the criteria listed in the tracked-application property.

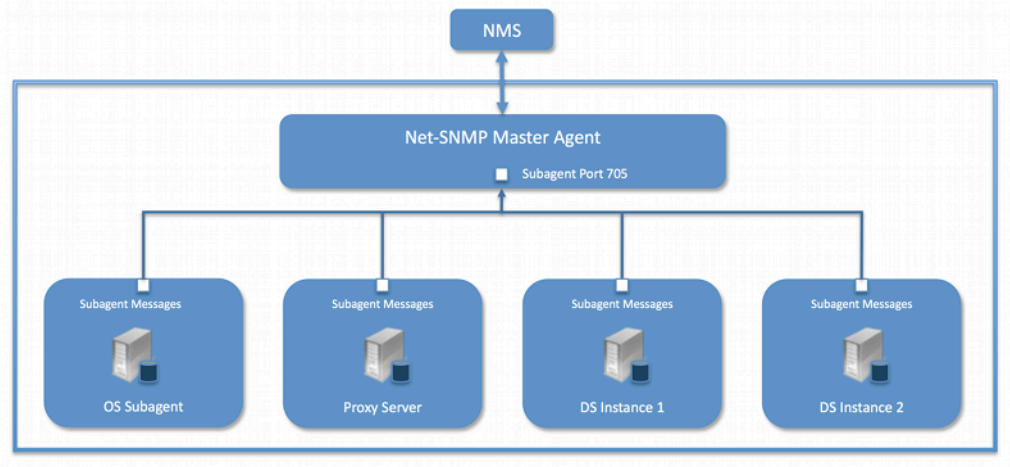
Monitoring using SNMP

The PingDirectoryProxy Server supports real-time monitoring using the Simple Network Management Protocol (SNMP). The Directory Proxy Server provides an embedded SNMPv3 subagent plugin that, when enabled, sets up the server as a managed device and exchanges monitoring information with a master agent based on the AgentX protocol.

SNMP implementation

In a typical SNMP deployment, many production environments use a network management system (NMS) for a unified monitoring and administrative view of all SNMP-enabled devices. The NMS communicates with a master agent, whose main responsibility is to translate the SNMP protocol messages and multiplex any request messages to the subagent on each managed device (for example, Directory Proxy Server instance, Directory Proxy Server, Data Sync Server, or OS Subagent). The master agent also processes responses or traps from the agents. Many vendors provide commercial NMS systems. Consult with your NMS system for specific information.

The PingDirectoryProxy Server contains an SNMP subagent plugin that connects to a Net-SNMP master agent over TCP. The main configuration properties of the plugin are the address and port of the master agent, which default to localhost and port 705, respectively. When the plugin is initialized, it creates an AgentX subagent and a managed object server, and then registers as a MIB server with the Directory Proxy Server instance. Once the plugin's startup method is called, it starts a session thread with the master agent. Whenever the connection is lost, the subagent automatically attempts to reconnect with the master agent. The Directory Proxy Server's SNMP subagent plugin transmits only read-only values for polling or trap purposes. (Set and inform operations are not supported). SNMP management applications cannot perform actions on the server on their own or by means of an NMS system.



MY TITLE Example SNMP Deployment

One important note is that the PingDirectoryProxy Server was designed to interface with a Net-SNMP (version 5.3.2.2 or later) master agent implementation with AgentX over TCP. Many operating systems provide their own Net-SNMP module. However, SMA disables some features present in the Net-SNMP package and only enables AgentX over UNIX Domain Sockets, which cannot be supported by Java. If your operating system has a native Net-SNMP master agent that only enables UNIX Domain Sockets, you must download and install a separate Net-SNMP binary from its web site.

Configuring SNMP

About this task

Because all server instances provide information for a common set of MIBs, each server instance provides its information under a unique SNMPv3 context name, equal to the server instance name. The server instance name is defined in the Global Configuration, and is constructed from the host name and the

server LDAP port by default. Consequently, information must be requested using SNMPv3, specifying the context name that pertains to the desired server instance. This context name is limited to 30 characters or less. Any context name longer than 30 characters will result in an error message. Since the default context name is limited to 30 characters or less, and defaults to the server instance name and the LDAP port number, pay special attention to the length of the fully-qualified (DNS) host name.

ⓘ **Note:** The Directory Proxy Server supports SNMPv3, and only SNMPv3 can access the MIBs. For systems that implement SNMP v1 and v2c, Net-SNMP provides a proxy function to route requests in one version of SNMP to an agent using a different SNMP version.

Steps

1. Enable the Directory Proxy Server's SNMP plugin by using the **`dsconfig`** tool. Make sure to specify the address and port of the SNMP master agent. On each Directory Proxy Server instance, enable the SNMP subagent. Note that the SNMPv3 context name is limited to 30 bytes maximum. If the default dynamically-constructed instance name is greater than 30 bytes, there will be an error when attempting to enable the plugin. Enable the SNMP Subagent Alert Handler so that the sub-agent will send traps for administrative alerts generated by the server.

   ```
   $ bin/dsconfig set-alert-handler-prop \
     --handler-name "SNMP Subagent Alert Handler" --set enabled:true
   ```

2. View the error log. You will see a message that the master agent is not connected, because it is not yet online.

   ```
   The SNMP sub-agent was unable to connect to the master
   agent at localhost/705: Timeout
   ```

3. Edit the SNMP agent configuration file, `snmpd.conf`, which is often located in `/etc/snmp/snmpd.conf`. Add the directive to run the agent as an AgentX master agent:

   ```
   master agentx agentXSocket tcp:localhost:705
   ```

   Note that the use of `localhost` means that only sub-agents running on the same host can connect to the master agent. This requirement is necessary since there are no security mechanisms in the AgentX protocol.

4. Add the trap directive to send SNMPv2 traps to `localhost` with the community name, public (or whatever SNMP community has been configured for your environment) and the port.

   ```
   trap2sink localhost public 162
   ```

5. To create a SNMPv3 user, add the following lines to the `/etc/snmp/snmpd.conf` file.

   ```
   rwuser initial
   createUser initial MD5 setup_passphrase DES
   ```

6. Run the following command to create the SNMPv3 user.

   ```
   snmpusm -v3 -u initial -n "" -l authNoPriv -a MD5 -A setup_passphrase \
   localhost create snmpuser initial
   ```

7. Start the **`snmpd`** daemon and after a few seconds you should see the following message in the Directory Proxy Server error log:

   ```
   The SNMP subagent connected successfully to the master agent
   at localhost:705. The SNMP context name is host.example.com:389
   ```

8. Set up a trap client to see the alerts that are generated by the Directory Proxy Server. Create a config file in `/tmp/snmptrapd.conf` and add the directive below to it. The directive specifies that the trap client can process traps using the public community string, and can log and trigger executable actions.

   ```
   authcommunity log, execute public
   ```

9. Install the MIB definitions for the Net-SNMP client tools, usually located in the `/usr/share/snmp/mibs` directory.

```
$ cp resource/mib/* /usr/share/snmp/mibs
```

10. Then, run the trap client using the **`snmptrapd`** command. The following example specifies that the command should not create a new process using **`fork()`** from the calling shell (`-f`), do not read any configuration files (`-C`) except the one specified with the `-c` option, print to standard output (`-Lo`), and then specify that debugging output should be turned on for the User-based Security Module (`-Dusm`). The path after the `-M` option is a directory that contains the MIBs shipped with our product (i.e., `server-root/resource/mib`) .

```
$ snmptrapd -f -C -c /tmp/snmptrapd.conf -Lf /root/trap.log -Dusm \
  -m all -M +/usr/share/snmp/mibs
```

11. Run the Net-SNMP client tools to test the feature. The following options are required: `-v <SNMP version>`, `-u <user name>`, `-A <user password>`, `-l <security level>`, `-n <context name (instance name)>` . The `-m` all option loads all MIBs in the default MIB directory in `/usr/share/snmp/mibs` so that MIB names can be used in place of numeric OIDs.

```
$ snmpget -v 3 -u snmpuser -A password -l authNoPriv -n host.example.com:389
 \
-m all localhost localDBBackendCount.0

$ snmpwalk -v 3 -u snmpuser -A password -l authNoPriv -n
 host.example.com:389 \
-m all localhost systemStatus
```

MIBS

The Directory Proxy Server provides SMIv2-compliant MIB definitions (RFC 2578, 2579, 2580) for distinct monitoring statistics. These MIB definitions are to be found in text files under `resource/mib` directory under the server root directory.

Each MIB provides managed object tables for each specific SNMP management information as follows:

- **LDAP Remote Server MIB**. Provides information related to the health and status of the LDAP servers that the Directory Proxy Server connects to, and statistics about the operations invoked by the Directory Proxy Server on those LDAP servers.
- **LDAP Statistics MIB**. Provides a collection of connection-oriented performance data that is based on a connection handler in the Directory Proxy Server. A server typically contain only one connection handler and therefore supplies only one table entry.
- **Local DB Backend MIB**. Provides key metrics related to the state of the local database backends contained in the server.
- **Processing Time MIB**. Provides a collection of key performance data related to the processing time of operations broken down by several criteria but reported as a single aggregated data set.
- **Replication MIB**. Provides key metrics related to the current state of replication, which can help diagnose how much outstanding work replication may have to do.
- **System Status MIB**. Provides a set of critical metrics for determining the status and health of the system in relation to its work load.

For information on the available monitoring statistics for each MIB available on the Directory Server and the Directory Proxy Server, see the text files provided in the `resource/mib` directory below the server installation.

The Directory Proxy Server generates an extensive set of SNMP traps for event monitoring. The traps display the severity, description, name, OID, and summary. For information about the available alert types for event monitoring, see the `resource/mib/UNBOUNDID-ALERT-MIB.txt` file.

Monitoring with the Administrative Console

About this task

Ping Identity has an Administrative Console for administrators to configure the directory server. The console also provides a status option that accesses the server's monitor content.

To view the Monitor Dashboard:

Steps

1. Ensure that the Directory Proxy Server is running.
2. Open a browser to `http://server-name:8443/console`.
3. Type the root user DN and password, and then click **Login**.
4. Use the top level navigation dropdown and select 'Status.'
5. On the Administrative Console's Status page, select the Monitors tab.

Accessing the Processing Time Histogram

About this task

The PingDirectoryProxy Server provides a processing time histogram that classifies operation response time into user-defined buckets. The histogram tracks the processing on a per operation basis and as a percentage of the overall processing time for all operations. It also provides statistics for each operation type (add, bind, compare, delete, modify, modify DN, search).

Steps

1. On the Administrative Console, click **Configuration > Status > Monitors tab**.
2. Select **Processing Time Histogram**. Other monitor entries can be accessed in similar ways.

Monitoring with JMX

The PingDirectoryProxy Server supports monitoring the JVM™ through a Java Management Extensions (JMX™) management agent, which can be accessed using JConsole or any other kind of JMX client. The JMX interface provides JVM performance and resource utilization information for applications running Java. You can monitor generic metrics exposed by the JVM itself, including memory pools, threads, loaded classes, and MBeans, as well as all the monitor information that the Directory Proxy Server provides. You can also subscribe to receive JMX notifications for any administrative alerts that are generated within the server.

Running JConsole

About this task

Before you can access JConsole, you must configure and enable the JMX Connection Handler for the Directory Proxy Server using the **`dsconfig`** tool. See Configuring the JMX Connection Handler and Alert Handler.

To invoke the JConsole executable, type `jconsole` on the command line. If *JDK_HOME* is not set in your path, you can access JConsole in the `bin` directory of the `JDK_HOME` path.

To run JConsole:

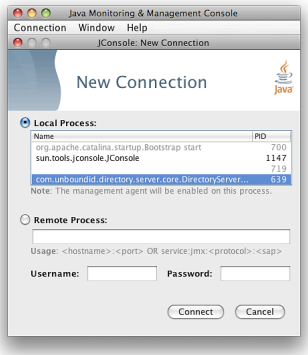
Steps

1. Use **`JConsole`** to open the Java Monitoring and Management Console. You can also run JConsole to monitor a specific process ID for your application: **`jconsole PID`**. Or you can run JConsole remotely using: **`jconsole hostname:port`**.

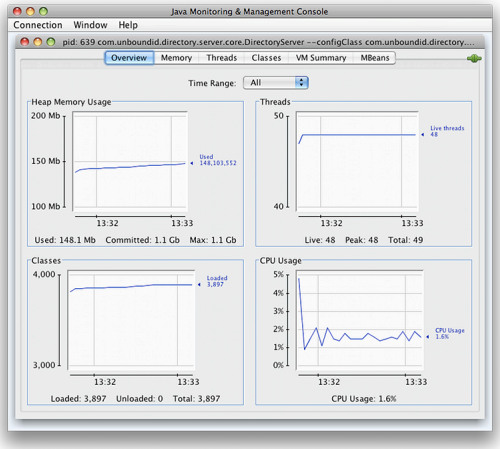
```
$ jconsole
```

> ⓘ **Note:** If SSL is configured on the JMX Connection Handler, you must specify the Directory Proxy Server jar file in the class path when running **`jconsole`** over SSL. For example, run the following **`jconsole`** command:
>
> ```
> $ jconsole \
>   -J-Djavax.net.ssl.trustStore=/path/to/certStores/truststore \
>   -J-Djavax.net.ssl.trustStorePassword=secret \
>   -J-Djava.class.path=$SERVER_ROOT/lib/PingDirectoryProxy.jar:/Library/Java/
> JavaVirtualMachines/jdk-version.jdk/Contents/Home/lib/jconsole.jar
> ```

2. On the **Java Monitoring & Administrative Console**, click **Local Process**, and then click the **PID** corresponding to the directory server.



3. Review the resource monitoring information.



Monitoring the Directory Proxy Server using JConsole

About this task

You can set up JConsole to monitor the Directory Proxy Server using a remote process. Make sure to enable the JMX Connection Handler and to assign at least the `jmx-read` privilege to a regular user

account (the `jmx-notify` privilege is required to subscribe to receive JMX notifications). Do not use a root user account, as this would pose a security risk.

Steps

1. Start the Directory Proxy Server.

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

2. Enable the JMX Connection handler using the **`dsconfig`** tool. The handler is disabled by default. Remember to include the LDAP connection parameters (host name, port, bindDN, bindPassword).

```
$ bin/dsconfig set-connection-handler-prop \
  --handler-name "JMX Connection Handler" --set enabled:true
```

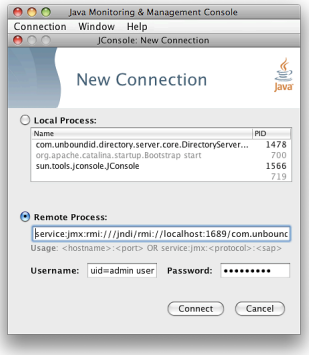
3. Assign `jmx-read`, `jmx-write`, and `jmx-notify` (if the user receives notifications) to the user.

```
$ bin/ldapmodify --hostname server1.example.com --port 1389 \
  --bindDN "cn=Directory Manager" --bindPassword secret
dn: uid=admin,dc=example,dc=com
changetype: modify
replace: ds-privilege-name
ds-privilege-name: jmx-read
ds-privilege-name: jmx-write
ds-privilege-name: jmx-notify
```

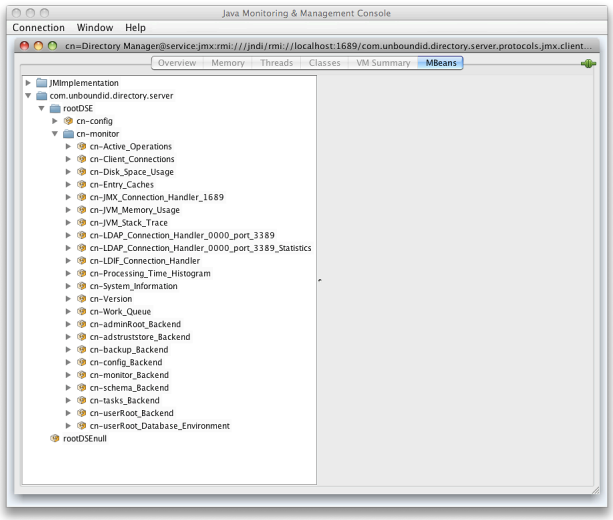
4. On the **Java Monitoring & Administrative Console**, click **Remote Process**, and enter the following JMX URL using the host and port of your Directory Proxy Server.

```
service:jmx:rmi:///jndi/rmi://<host>:<port>/
com.unboundid.directory.server.protocols.jmx.client-unknown
```

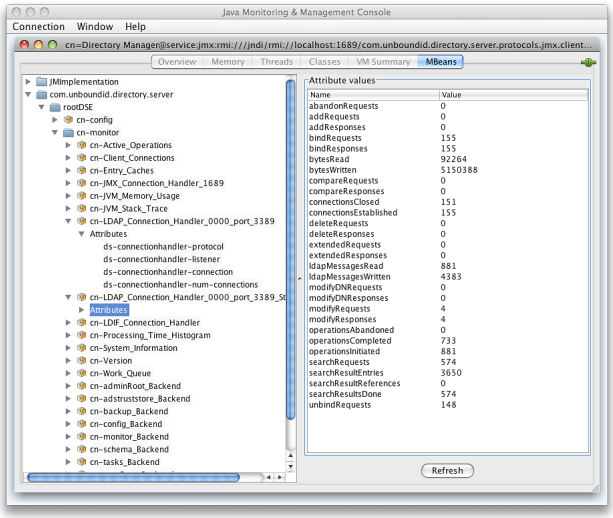
5. In the **Username** and **Password** fields, type the bind DN and password for a user that has at least the `jmx-read` privilege. Click **Connect**.



6. Click **com.unboundid.directory.server**, and expand the `rootDSE` node and the `cn-monitor` sub-node.



7. Click a monitoring entry. In this example, click the **LDAP Connection Handler** entry.



Monitoring Using the LDAP SDK

You can use the monitoring API to retrieve monitor entries from the Directory Proxy Server as well as to retrieve specific types of monitor entries.

For example, you can retrieve all monitor entries published by the Directory Proxy Server and print the information contained in each using the generic API for accessing monitor entry data as follows:

```
for (MonitorEntry e : MonitorManager.getMonitorEntries(connection))
  {
    System.out.println("Monitor Name: " + e.getMonitorName());
    System.out.println("Monitor Type: " + e.getMonitorDisplayName());
    System.out.println("Monitor Data:");
    for (MonitorAttribute a : e.getMonitorAttributes().values())
    {
      for (Object value : a.getValues())
      {
        System.out.println(" " + a.getDisplayName() + ": " +
 String.valueOf(value));
      }
```

```
      }
      System.out.println();
   }
```

For more information about the LDAP SDK and the methods in this example, see the *LDAP SDK* documentation.

Monitoring over LDAP

The PingDirectoryProxy Server exposes a majority of its information under the `cn=monitor` entry. You can access these entries over LDAP using the **`ldapsearch`** tool.

```
$ bin/ldapsearch --hostname server1.example.com --port 1389 \
  --bindDN "uid=admin,dc=example,dc=com" --bindPassword secret \
  --baseDN "cn=monitor" "(objectclass=*)"
```

Profiling Server Performance Using the Stats Logger

The Directory Proxy Server ships with a built-in Stats Logger that is useful for profiling server performance for a given configuration. At a specified interval, the Stats Logger can write server statistics to a JSON file or to a log file in a comma-separated format (.csv), which can be read by spreadsheet applications. The logger has a negligible impact on server performance unless the `log-interval` property is set to a very small value (less than 1 second). The statistics logged and their verbosity can be customized.

The Stats Logger can also be used to view historical information about server statistics including replication, LDAP operations, host information, and gauges. Either update the configuration of the existing Stats Logger Plugin to set the advanced `gauge-info` property to `basic/extended` to include this information, or create a dedicated Periodic Stats Logger for information about statistics of interest.

For information about how to enable this logging and where to find the log file, see *To Enable the Stats Logger* on page 755.

To Enable the Stats Logger

About this task
By default, the Directory Proxy Server ships with the built-in "Stats Logger' disabled. To enable it using the **`dsconfig`** tool or the Administrative Console, go to **Plugins** menu (available on the Advanced object menu), and then, select .

Steps

1. Run **`dsconfig`** in interactive mode. Enter the LDAP or LDAPS connection parameters when prompted.

   ```
   $ bin/dsconfig
   ```
2. Enter `o` to change to the Advanced Objects menu.
3. On the main menu, enter the number for Plugins.
4. On the **Plugin** menu, enter the number corresponding to view and edit an existing plugin.
5. On the **Plugin selection** list, enter the number corresponding to the Stats Logger.
6. On the **Stats Logger Plugin** menu, enter the number to set the `enabled` property to `TRUE`. When done, enter `f` to save and apply the configuration. The default logger will log information about the server every second to `<server-root>/logs/dsstats.csv`. If the server is idle, nothing will be logged, but this can be changed by setting the `suppress-if-idle` property to `FALSE` (`suppress-if-idle=false`).

   ```
   >>>> Configure the properties of the Stats Logger Plugin

   Property                Value(s)
   ----------------------------------------------------------------------
   ```

```
1)   description          Logs performance stats to a log file
                         periodically.
2)   enabled                        false
3)   local-db-backend-info          basic
4)   replication-info               basic
5)   entry-cache-info               -
6)   host-info                      -
7)   included-ldap-application      If per-application LDAP stats is enabled,
                                    then stats will be included for all
                                    applications.
8)   log-interval                   1 s
9)   collection-interval            200 ms
10)  suppress-if-idle               true
11)  header-prefix-per-column       false
12)  empty-instead-of-zero          true
13)  lines-between-header           50
14)  included-ldap-stat             active-operations, num-connections,
                                    op-count-and-latency, work-queue
15)  included-resource-stat         memory-utilization
16)  histogram-format               count
17)  histogram-op-type              all
18)  per-application-ldap-stats     aggregate-only
19)  ldap-changelog-info            -
20)  gauge-info                     none
21)  log-file                       logs/dsstats.csv
22)  log-file-permissions           640
23)  append                         true
24)  rotation-policy                Fixed Time Rotation Policy, Size Limit
                                    Rotation Policy
25)  retention-policy               File Count Retention Policy

?)   help
f)   finish - apply any changes to the Periodic Stats Logger Plugin
a)   hide advanced properties of the Periodic Stats Logger Plugin
d)   display the equivalent dsconfig command lines to either re-create this
          object or only to apply pending changes
b)   back
q)   quit

Enter choice [b]:
```

7. Run the Directory Proxy Server. For example, if you are running in a test environment, you can run the **`search-and-mod-rate`** tool to apply some searches and modifications to the server. You can run **`search-and-mod-rate`** `--help` to see an example command.
8. View the Stats log output at `<server-root>/logs/dsstats.csv`. You can open the file in a spreadsheet.

To Configure Multiple Periodic Stats Loggers

About this task

Multiple Periodic Stats Loggers can be created to log different statistics, view historical information about gauges, or to create a log at different intervals (such as logging cumulative operations statistics every hour). To create a new log, use the existing Stats Logger as a template to get reasonable settings, including rotation and retention policy.

Steps

1. Run **`dsconfig`** by repeating steps 1–3 in *To Enable the Stats Logger*.
2. From the **Plugin management** menu, enter the number to create a new plugin.

3. From the **Create a New Periodic Stats Logger Plugin** menu, enter `t` to use an existing plugin as a template.
4. Enter the number corresponding to the existing stats logger as a template.
5. Next, enter a descriptive name for the new stats logger. For this example, type `Stats Logger-10s`.
6. Enter the log file path to the file. For this example, type `logs/dsstats2.csv`.
7. On the menu, make any other change to the logger. For this example, change the `log-interval` to 10s, and the `suppress-if-idle` to false. When finished, enter `f` to save and apply the configuration.
8. You should now see two loggers `dsstats.csv` and `dsstats2.csv` in the `logs` directory.

Adding custom logged statistics to a Periodic Stats Logger

Add custom statistics based on any attribute in any entry under `cn=monitor` using the Custom Logged Stats object. This configuration object provides powerful controls for how monitor attributes are written to the log. For example, you can extract a value from a monitor attribute using a regular expression. Newly created Custom Logged Stats will automatically be included in the Periodic Stats Logger output.

Besides allowing a straight pass-through of the values using the 'raw' statistic-type, you can configure attributes to be treated as a counter (where the interval includes the difference in the value since the last interval), an average, a minimum, or a maximum value held by the attribute during the specified interval. The value of an attribute can also be scaled by a fixed value or by the value of another monitor attribute.

ⓘ **Note:** Custom third-party server extensions that were written using the Server SDK can also expose interval statistics using a Periodic Stats Logger. The extension must first implement the SDK's `MonitorProvider` interface and register with the server. The monitor attributes produced by this custom `MonitorProvider` are then available to be referenced by a Custom Logged Stats object.

To illustrate how to configure a Custom Logged Statistics Logger, the following procedure reproduces the built-in "Consumer Total GB" column that shows up in the output when the `included-resource-stat` property is set to memory-utilization on the Periodic Stats Logger. The column is derived from the `total-bytes-used-by-memory-consumers` attribute of the `cn=JVM Memory Usage,cn=monitor` entry as follows:

```
dn: cn=JVM Memory Usage,cn=monitor
objectClass: top
objectClass: ds-monitor-entry
objectClass: ds-memory-usage-monitor-entry
objectClass: extensibleObject
cn: JVM Memory Usage
...
total-bytes-used-by-memory-consumers: 3250017037
```

Configuring a custom logged statistic using dsconfig interactive

Steps

1. Run **`dsconfig`** and enter the LDAP/LDAPS connection parameters when prompted.

   ```
   $ bin/dsconfig
   ```
2. On the Directory Proxy Server configuration main menu (Advanced Objects menu), enter the number corresponding to Custom Logged Stats.
3. On the Custom Logged Stats menu, enter the number corresponding to Create a new Custom Logged Stats.
4. Select the Stats Logger Plugin from the list if more than one is present on the system. If you only have one stats logger, press **Enter** to confirm that you want to use the existing plugin.
5. Enter a descriptive name for the Custom Logged Stats. For this example, enter `Memory Usage`.

6. From the `monitor-objectclass` property menu, enter the objectclass attribute to monitor. For this example, enter `ds-memory-usage-monitor-entry`. You can run **`ldapsearch`** using the base DN "cn=JVM Memory Usage,cn=monitor" entry to view the entry.
7. Next, specify the attributes of the monitor entry that you want to log in the stats logger. In this example, enter `total-bytes-used-by-memory-consumers`, and then press **Enter** again to continue.
8. Next, specify the type of statistics for the monitored attribute that will appear in the log file. In this example, enter the option for raw statistics as recorded by the logger.
9. In the Custom Logged Stats menu, review the configuration. At this point, we want to set up a column name that lists the Memory Usage. Enter the option to change the `column-name` property.
10. Next, we want to add a specific label for the column name. Enter the option to add a value, and then enter **Memory Consumer Total (GB)**, and press **Enter** again to continue.
11. Confirm that you want to use the `column-name` value that you entered in the previous step, and then press `Enter` to use the value.
12. Next, we want to scale the Memory Consumer Totals by one gigabyte. On the **Custom Logged Stats** menu, enter the option to change the `divide-value-by` property.
13. On the `divide-value-by` property menu, enter the option to change the value, and then enter 1073741824 (i.e., 1073741824 bytes = 1 gigabytes).
14. On the **Custom Logged Stats** menu, review your configuration. When finished, enter `f` to save and apply the settings.

```
>>>> Configure the properties of the Custom Logged Stats
 >>>> via creating 'Memory Usage' Custom Logged Stats

         Property                     Value(s)
         ---------------------------------------------------------------
    1)   description                  -
    2)   enabled                      true
    3)   monitor-objectclass          ds-memory-usage-monitor-entry
    4)   include-filter               -
    5)   attribute-to-log             total-bytes-used-by-memory-consumers
    6)   column-name                  Memory Consumer Total (GB)
    7)   statistic-type               raw
    8)   header-prefix                -
    9)   header-prefix-attribute      -
    10)  regex-pattern                -
    11)  regex-replacement            -
    12)  divide-value-by              1073741824
    13)  divide-value-by-attribute    -
    14)  decimal-format               #.##
    15)  non-zero-implies-not-idle    false

    ?)   help
    f)   finish - create the new Custom Logged Stats
    a)   hide advanced properties of the Custom Logged Stats
    d)   display the equivalent dsconfig arguments to create this object
    b)   back
    q)   quit

Enter choice [b]:
```

```
The Custom Logged Stats was created successfully
```

When the Custom Logged Stats configuration change is completed, the new stats value should immediately show up in the Stats Logger output file.

Configuring a custom stats logger using dsconfig non-interactive

Steps

- Use the **dsconfig** non-interactive command-line equivalent to create your custom stats logger. The following one-line command replicates the procedure in the previous section. This command produces a column named "Memory Consumer Total (GB)" that contains the value of the of `total-bytes-used-by-memory-consumers` attribute pulled from the entry with the `ds-memory-usage-monitor-entry` objectclass. This value is scaled by 1073741824 to get to a value represented in GBs.

```
$ bin/dsconfig create-custom-logged-stats --plugin-name "Stats Logger" \
  --stats-name "Memory Usage" --type custom \
  --set monitor-objectclass:ds-memory-usage-monitor-entry \
  --set attribute-to-log:total-bytes-used-by-memory-consumers \
  --set "column-name:Memory Consumer Total (GB)" --set statistic-type:raw \
  --set divide-value-by:1073741824
```

StatsD monitoring endpoint

The Monitoring Endpoint configuration type provides the StatsD Endpoint type that you can use to transfer metrics data in the StatsD format.

Examples of metrics you can send are:

- Busy worker thread count
- Garbage collection statistics
- Host system metrics such as CPU and memory

For a list of available metrics, use the interactive **dsconfig** menu for the Stats Collector Plugin or in the Administrative Console, edit the **Stats Collector** plugin as explained in the second example.

You configure the Monitoring Endpoint using the **dsconfig** command. When you configure Monitoring Endpoint, you include:

- The endpoint's hostname
- The endpoint's port
- A toggle to use TCP or UDP
- A toggle to use SSL if you use TCP

For example, to configure a new StatsD Monitoring Endpoint to send UDP data to localhost port 8125 using **dsconfig**:

```
dsconfig create-monitoring-endpoint \
    --type statsd \
    --endpoint-name StatsDEndpoint \
    --set enabled:true \
    --set hostname:localhost \
    --set server-port:8125 \
    --set connection-type:unencrypted-udp
```

If you are using the Administrative Console:

1. Click **Show Advanced Configuration**.
2. In the **Logging, Monitoring, and Notifications** section, click **Monitoring Endpoints**.
3. Click **New Monitoring Endpoint**.

You can send data to any number of monitoring endpoints.

The Stats Collector Plugin controls the metrics used by the StatsD monitoring endpoint. To send metrics with the StatsD monitoring endpoint, you must enable the Stats Collector Plugin. Also, you must configure the Stats Collector Plugin to indicate the metrics to send.

To enable the Stats Collector Plugin or to configure the type of data sent, use the **dsconfig** command or the Administrative Console. For example, to enable the Stats Collector Plugin to send host CPU metric, memory metrics, and server status metrics using **dsconfig**:

```
dsconfig set-plugin-prop \
    --plugin-name "Stats Collector" \
    --set enabled:true \
    --set host-info:cpu \
    --set host-info:disk \
    --set status-summary-info:basic
```

If you are not using Data Metrics Server to monitor your server, you can disable the generation of some metrics files that are not necessary for the StatsD Monitoring Endpoint. To do this, set the `generate-collector-files` property on the Stats Collector Plugin to `false`.

If you are using the Administrative Console:

1. Click **Show Advanced Configuration**.
2. In the **LDAP (Administration and Monitoring)** section, click **Plugin Root**
3. Edit the **Stats Collector** plugin.

After you enable the Stats Collector and create the StatsD monitoring endpoint, you can:

- Use the data with Splunk as explained in *Sending Metrics to Splunk with StatsD* on page 758.
- Configure other tools that support StatsD, such as CloudWatch or a Prometheus StatsD exporter, to use the data. For more information about this configuration, see your tool's StatsD documentation. Configure the StatsD monitoring endpoint to use the correct host and port. The `dsconfig create-monitoring-endpoint` example above uses a host of localhost and a port of 8125. You can also set these values in the Administrative Console.

Sending Metrics to Splunk with StatsD

About this task

With the StatsD Endpoint type, you can send metric data to a Splunk installation.

In Splunk, you can use SSL to secure ports that are open for StatsD.

> ⓘ **Note:**
>
> StatsD metrics are typically sent over UDP. By using UDP, the client sending metrics does not have to block as it would if using TCP. However, using TCP guarantees order and ensures no metrics are lost.

You can configure open UDP (or TCP) ports in Splunk to accept only connections from a certain hostname or IP address.

To securely send UDP (or TCP) data to Splunk, you can:

Steps

1. Send the data to a Splunk Universal Forwarder.
2. Have the forwarder communicate with the Splunk Indexer over SSL.

DevOps and Infrastructure as Code

Derived from the words *development* and *operations*, the term *DevOps* refers to the practices that a company follows to ensure the production of high-quality products, while also minimizing the amount of time between the commitment of a system change and the implementation of that change in a production environment.

Most companies practice one of the following service models:

- Pets — With the *pets service model*, servers are built and managed manually, and are treated as unique and indispensable. Examples include mainframes, database systems, and load balancers.
- Cattle — With the *cattle service model*, arrays of multiple replaceable servers are built with automated tools. During a failure event, an array automatically restarts failed services and replicates data. Examples include web server arrays, search clusters, and multi-master datastores.

Historically, servers have been treated like pets. The failure of one or multiple servers was often viewed as an emergency, and extensive resources were usually required to repair the damage. In the new DevOps paradigm, servers are recognized as dispensable and treated like cattle. When a server fails, the infrastructure replaces it immediately, configuring the replacement server identically to the failed one. Because no human intervention is required to fix them, the servers are considered self-healing.

To help treat your servers more like cattle than pets, PingDirectoryProxy Server supports server profiles and features like topology-management tools. For example, the **`manage-profile setup`** represents a single command that performs all the steps from the pet service model on a `server-profile` directory, a well-defined directory structure with all the necessary server configuration bits. Similarly, the **`manage-profile replace-profile`** command performs similar steps with a single command invocation after a server is updated to a new version.

The scripts in the `server-profile` directory are declarative of the environment. Consequently, what you define in the `server-profile` directory is what you get on the servers. No one needs to identify a server's current configuration and compute the differences that must be applied to attain the appropriate end state. Prior to server profiles, this problem was difficult to address, especially where no history was available. In such scenarios, an administrator might have needed to obtain the current configuration from the servers, to manually compute the difference between the current and desired configuration, and to apply the configuration changes, hoping all the while that nobody had changed the configuration during the process. Server profiles eliminate this procedural approach to applying configuration changes, and they simplify the steps associated with determining what is deployed in an environment. For more information on server profiles, see *Server profiles* on page 816.

Another principle that relates closely to DevOps is infrastructure as code (IaC), the concept of managing your operations in the same manner as your application and other code for general release with proper versioning, continuous integration, quality control, and release cycles. Customers who deploy PingDirectoryProxy Servers as pets today can take advantage of current DevOps and IaC principles to turn them into cattle.

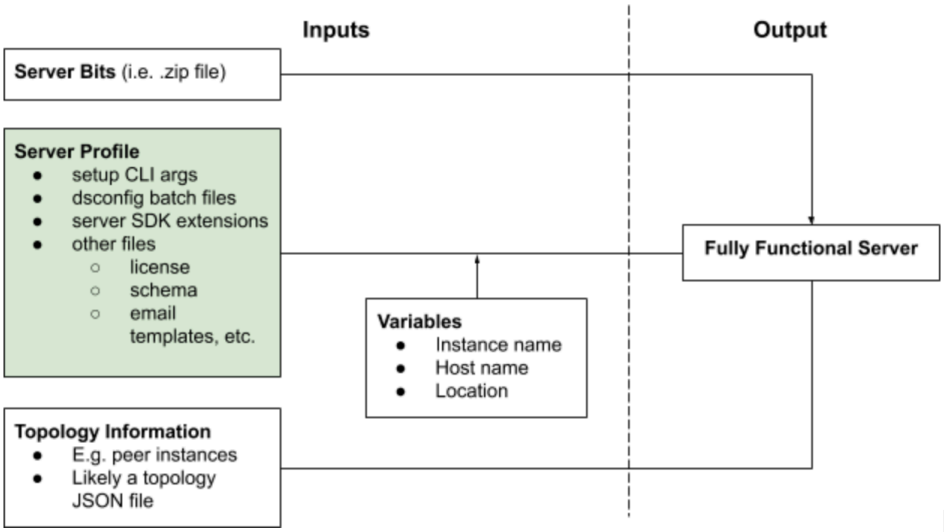
Server profiles

Regardless of the service model that your company follows, *server profiles* help you achieve your goals. At a basic level, a server profile defines a format for the configuration of a server by combining the following files into a single concrete structure:

- **`dsconfig`**
- Setup arguments
- Server SDK extensions
- Additional miscellaneous files

The primary goal of a server profile is to simplify the deployment of PingDirectoryProxy Server and related products by using deployment automation frameworks. When products support this capability, the amount of scripting that is required across automation frameworks — like Docker, Kubernetes, and Ansible — is reduced considerably.

The following image shows the role that a server profile plays in building a fully functional running server.



As a declarative form of a full server configuration, a server profile provides the following advantages:

- Provides a more complete and easily comparable way to define an individual server's configuration. Changes between different servers are easier to understand, and incremental changes to a server's configuration are easier to track.
- Ensures that each server instance is configured identically to its peers.
- Can be applied directly to new as well as previously installed instances.
- Shares a common configuration across a deployment pipeline of development, test, and production environments without unnecessary duplication. For information about substituting variables that differ by environment, see *Variable substitution* on page 817.
- Facilitates deployment automation by representing configuration as code.
- Reduces the number of additional configuration steps that are required to place a server into production.
- Makes the execution of various configuration changes more consistent and repeatable. The strategy of using a server profile to represent the final state of a server is less error-prone than recording a step-by-step process to attain that state.
- Can be managed easily in a version-control system.
- Simplifies the management of servers outside deployment automation frameworks.

A continuous deployment workflow can work with server profiles to make certain that changes to a server profile are moved automatically into production. In a stateful environment, the **`manage-profile replace-profile`** subcommand can be used to update existing servers. In a stateless environment, in which servers are considered immutable, **`manage-profile setup`** can be used to deploy new servers whenever a profile changes. With multiple environments, this deployment can be performed in a test environment before moving to production.

When working with zipped server SDK extensions and other files that might not be stored in a version-control system, the server profile can be modified to include these files prior to its use. For example, if the code for an extension is stored in a separate repository, it can be built and dropped into the server profile immediately before the **`manage-profile`** tool is run. This process is part of the deployment automation logic that uses the server profile, and it can be followed for any files that are needed by the server profile but whose versions are not controlled.

For more information about the **`manage-profile`** tool, see *About the manage-profile tool* on page 820.

Variable substitution

You can use the **manage-profile** tool to substitute different variables in server profiles.

The **manage-profile** tool uses the format **${VARIABLE}** to support the substitution of variables in profiles. This format can be escaped by using another **$**. For example, after substitution, **$${VARIABLE}** becomes **${VARIABLE}**.

Variable values can be read from a profile variables file or from environment variable values. If both options are used, the values that are specified in the file overwrite any environment variables.

The following code provides an example of how you can set user-defined variables by using a variables file in the server profile.

```
HOSTNAME=testserver.example.com
PORT=389
```

The following table describes built-in variables that can also be referenced in the server profile. Use these variables in the format previously described.

Built-in variable	Description
PING_SERVER_ROOT	Evaluates to the absolute path of the server's root directory
PING_PROFILE_ROOT	Evaluates to the individual profile's root directory ⓘ **Note:** Use PING_PROFILE_ROOT only with files that are not needed after initial setup, such as password files in `setup-arguments.txt`. Do not use the PING_PROFILE_ROOT variable for files needed while the server is running. The **manage-profile** tool creates a temporary copy of the server profile that is deleted after the tool completes, so files are not accessible under PING_PROFILE_ROOT when the server is running. For files you need while the server is running, such as keystore and truststore files, copy the files into the server root using the profile's `server-root/pre-setup` directory, and then refer to the files using with the PING_SERVER_ROOT variable.

For more information about the tool's usage, run the command **bin/manage-profile --help**.

Profile Structure

Use either of the following methods to create a server profile:

- Use the template named `server-profile-template.zip`, which is located in the `resource/` directory.
- Run the **manage-profile generate-profile** subcommand. The **manage-profile** tool references a file system directory structure rather than a ZIP file.

Files can be added to each directory as needed.

The following hierarchy represents the file structure of a basic server profile:

```
-server-profile/
  |-- dsconfig/
```

```
|-- misc-files/
|-- server-root/
|   |-- post-setup/
|   |-- pre-setup/
|-- server-sdk-extensions/
|-- setup-arguments.txt
|-- variables-ignore.txt
```

setup-arguments.txt

When creating a profile, the first step is to add arguments to the file `setup-arguments.txt`. When **`manage-profile`** setup is run, these arguments are passed to the server's setup tool. To view the arguments that are available in this file, run the server's **`setup --help`** command.

To provide the equivalent, non-interactive CLI arguments after any prompts have been completed, run **`setup`** interactively. The `setup-arguments.txt` file in the profile template contains an example set of arguments that can be changed.

`setup-arguments.txt` is the only required file in the profile.

dsconfig/

`dsconfig` batch files can be added to the `dsconfig` directory. These files, each of which must include a `.dsconfig` extension, contain **`dsconfig`** commands to apply to server.

Because the **`dsconfig`** batch files are ordered lexicographically, `00-base.dsconfig` runs `before 01-second.dsconfig`, and so on.

To produce a `dsconfig` batch file that reproduces the current configuration, run **`bin/config-diff`**.

server-root/

Any server root files can be added to the `server-root` directory, including schema files, email template files, custom password dictionaries, and other files that must be present on the final server root. Add these files to the `server-root/pre-setup` or `server-root/post-setup` directory, depending on when they need to be copied to the server root. Most server root files are added to the server-root/pre-setup directory.

server-sdk-extensions/

Add server SDK extension `.zip` files to the `server-sdk-extensions` directory. Include any configuration that is necessary for the extensions in the profile's **`dsconfig`** batch files.

variables-ignore.txt

The `variables-ignore.txt` file is an optional component of the server profile. It is useful when adding bash scripts to the server root because such files often contain expressions that the **`manage-profile`** tool normally interprets as variables.

Add `variables-ignore.txt` to a profile's root directory to indicate the relative paths of any files that are not to have their variables substituted.

The following example shows the contents of a typical `variables-ignore.txt` file:

```
server-root/pre-setup/script-to-ignore.sh
server-root/post-setup/another-file-to-ignore.txt
```

server-root/permissions.properties

The `permissions.properties` file, located in the `server-root` directory, is an optional file that specifies the permissions to apply to files that are copied to the server root. These permissions are represented in octal notation. By default, server root files maintain their permissions when copied.

The following example shows the contents of a typical `permissions.properties` file:

```
default=700
file-with-special-permissions.txt=600
new-subdirectory/file-with-special-permissions.txt=644
bin/example-script.sh=760
```

misc-files/

Additional documentation and other files can be added to the `misc-files` directory, which the **`manage-profile`** tool does not use. Use the variable **`PING_PROFILE_ROOT`** to refer to files in this directory from other locations, such as `setup-arguments.txt`.

> ⓘ **Note:**
>
> Use PING_PROFILE_ROOT only with files that are not needed after initial setup, such as password files in `setup-arguments.txt`. Do not use the PING_PROFILE_ROOT variable for files needed while the server is running. The **`manage-profile`** tool creates a temporary copy of the server profile that is deleted after the tool completes, so files are not accessible under PING_PROFILE_ROOT when the server is running. For files you need while the server is running, such as keystore and truststore files, copy the files into the server root using the profile's `server-root/pre-setup` directory, and then refer to the files using with the PING_SERVER_ROOT variable.

For example, a password file named `password.txt` in the `misc-files` directory could be referenced with `${PING_PROFILE_ROOT}/misc-files/password.txt` in `setup-arguments.txt`. Use a reference like this example to supply the file for the **`--rootUserPasswordFile`** argument in `setup-arguments.txt`.

About the manage-profile tool

The **`manage-profile`** tool is provided with the server to work with server profiles. It includes subcommands for creating, applying, and replacing server profiles, all of which significantly reduce the effort required by an administrator to configure a server appropriately.

The following sections describe these subcommands in more detail. For more information about the **`manage-profile`** tool, run **`manage-profile --help`**. For more information about each individual subcommand and its options, run **`manage-profile <subcommand> --help`**.

manage-profile generate-profile

To create a server profile from a configured server, use the **`generate-profile`** subcommand. The generated profile contains the following information, which provides a base for completing a profile:

- Command-line arguments that were used to set up the server
- **`dsconfig`** commands necessary to configure the server
- Installed server SDK extensions
- Files that are added to the server root

To produce a complete profile, some parts of the generated profile might require modifications, such as adding password files that `setup-arguments.txt` uses. The **`--instanceName`** and **`--localHostName`** arguments in `setup-arguments.txt` are made variables by **`generate-profile`**, and must be provided values when other **`manage-profile`** subcommands use the generated profile.

manage-profile setup

To apply a server profile to a fresh, unconfigured server, use the **`setup`** subcommand, which replaces the normal setup tool when using a server profile. Run **`manage-profile setup`** to complete the following tasks:

- Copies the pre-setup files to the server root
- Runs the setup tool

- Copies the post-setup files to the server root
- Installs any server SDK extensions
- Runs any dsconfig batch files
- Imports any LDIF files
- Installs the server SDK extensions
- Starts the server

While manage-profile setup is running, a copy of the profile is created in a temporary directory that can be specified by using the **--tempProfileDirectory** argument. The command leaves the server in a complete and running state when finished, unless the **--doNotStart** argument is specified.

manage-profile replace-profile

Run the **replace-profile** subcommand on a server that was originally set up with a server profile to replace its configuration with a new profile. The tool applies a specified server profile to an existing server while preserving its data, topology configuration, and replication configuration.

While **manage-profile replace-profile** is running, the existing server is stopped and moved to a temporary directory that the **--tempServerDirectory** argument can specify. A fresh, new server is subsequently installed and set up with the new profile. The final server is left running if it was running before the command was started, and remains stopped if it was stopped.

Run **manage-profile replace-profile** from a second unzipped server install package on the same host as the existing server, similar to the **update** tool. Use the **--serverRoot** argument to specify the root of the existing server that will have its profile replaced.

If files have been added or modified in the server root since the most recent **manage-profile setup** or **manage-profile replace-profile** was run, they are included in the final server with the replaced profile. Otherwise, files specifically added from the `server-root` directory of the previous server profile are absent from the final server with the replaced profile. If errors occur during the subcommand, such as the new profile having an invalid `setup-arguments.txt` file, the existing server returns to its original state from before **manage-profile replace-profile** was run.

ⓘ **Note:** The **manage-profile replace-profile** tool can update the server version when needed.

ⓘ **Note:** The **manage-profile replace-profile** tool can directly apply configuration changes when there are no other changes in the new profile. This is a shorter process when making small changes to **dsconfig**.

Server Profiles in a Pets Service Model

Server profiles and other DevOps concepts are also invaluable in a pets service model. For example, the step of using the **manage-profile generate-profile** subcommand to generate a server profile from a production server creates an easily consumable representation of the server's configuration. In nearly every scenario, the generation of a profile from an existing server is simpler than the piecing together manually of schemas, extensions, and other configuration information to create an image of that server. Additionally, generated profiles can be backed up or checked in to source control to maintain a consistent picture of an active server's configuration.

Another valuable use of server profiles involves setting up servers in a test environment that is separate from production. For example, a profile that matches the profile of a production server can be generated and used to install a fresh test server that matches the production server. Further, variable substitution allows environmental changes, such as local host name or instance name, without requiring a separate profile. Because the server's original configuration matches the running production server, adjustments can be tested easily. This approach provides more consistency when you validate changes before moving them to production.

If a new pet server has been set up with a server profile, **manage-profile replace-profile** can be used to apply changes to the profile. Rather than using scripts or a manual process to apply individual

changes, **replace-profile** provides a consistent, repeatable method of moving to a new server profile. This strategy automates more easily and is less prone to human error.

For more information about the **manage-profile** tool, see *About the manage-profile tool* on page 820.

Troubleshooting the Directory Proxy Server

This chapter provides the common problems and potential solutions that might occur when running PingDirectoryProxy Server. It is primarily targeted at cases in which the Directory Proxy Server is running on Linux® systems, but much of the information can be useful on other platforms as well.

This chapter presents the following information:

Garbage Collection Diagnostic Information

You can enable the JVM debugging options to track garbage collection data for your system. The options can impact JVM performance, but they provide valuable data to tune your server when troubleshooting garbage collection issues. While the **jstat** utility with the `-gc` option can be used to obtain some information about garbage collection activity, there are additional arguments that can be added to the JVM to use when running the server to provide additional detail.

```
-XX:+PrintGCDetails
-XX:+PrintTenuringDistribution
-XX:+PrintGCApplicationConcurrentTime
-XX:+PrintGCApplicationStoppedTime
-XX:+PrintGCDateStamps
```

To run the Directory Proxy Server with these options, edit the `config/java.properties` file and add them to the end of the line that begins with "`bin/start-server.java-args`". After the file has been saved, invoke the following command to make those new arguments take effect the next time the server is started:

```
$ bin/dsjavaproperties
```

Working with the Troubleshooting Tools

If problems arise with the Directory Proxy Server (whether from issues in the Directory Proxy Server itself or a supporting component, like the JVM, operating system, or hardware), then it is essential to be able to diagnose the problem quickly to determine the underlying cause and the best course of action to take towards resolving it.

Working with the collect-support-data tool

The Directory Proxy Server provides a significant amount of information about its current state including any problems that it has encountered during processing. If a problem occurs, the first step is to run the **collect-support-data** tool in the `bin` directory. The tool aggregates all relevant support files into a zip file that administrators can send to your authorized support provider for analysis. The tool also runs data collector utilities, such as **jps**, **jstack**, and **jstat** plus other diagnostic tools, and bundles the results in the zip file.

The tool may only archive portions of certain log files to conserve space, so that the resulting support archive does not exceed the typical size limits associated with e-mail attachments.

The data collected by the **collect-support-data** tool varies between systems. However, the tool always tries to get the same information across all systems for the target Directory Proxy Server. The data collected includes the configuration directory, summaries and snippets from the `logs` directory, an LDIF of the monitor and RootDSE entries, and a list of all files in the server root.

Available tool options

The **`collect-support-data`** tool has some important options that you should be aware of:

- **--noLdap**. Specifies that no effort should be made to collect any information over LDAP. This option should only be used if the server is completely unresponsive or will not start and only as a last resort.
- **--pid {pid}**. Specifies the ID of an additional process from which information is to be collected. This option is useful for troubleshooting external server tools and can be specified multiple times for each external server, respectively.
- **--sequential**. Use this option to diagnose “Out of Memory” errors. The tool collects data in parallel to minimize the collection time necessary for some analysis utilities. This option specifies that data collection should be run sequentially as opposed to in parallel. This action has the effect of reducing the initial memory footprint of this tool at a cost of taking longer to complete.
- **--reportCount {count}**. Specifies the number of reports generated for commands that supports sampling (for example, **`vmstat`**, **`iostat`**, or **`mpstat`**). A value of 0 (zero) indicates that no reports will be generated for these commands. If this option is not specified, it defaults to 10.
- **--reportInterval {interval}**. Specifies the number of seconds between reports for commands that support sampling (for example, **`mpstat`**). This option must have a value greater than 0 (zero). If this option is not specified, it default to 1.
- **--maxJstacks {number}**. Specifies the number of jstack samples to collect. If not specified, the default number of samples collected is 10.
- **--collectExpensiveData**. Specifies that data on expensive or long running processes be collected. These processes are not collected by default, because they will impact the performance of a running server.
- **--comment {comment}**. Provides the ability to submit any additional information about the collected data set. The comment will be added to the generated archive as a README file.
- **--includeBinaryFiles**. Specifies that binary files be included in the archive collection. By default, all binary files are automatically excluded in data collection.
- **--adminPassword {adminPassword}**. Specifies the global administrator password used to obtain **`dsreplication status`** information.
- **--adminPasswordFile {adminPasswordFile}**. Specifies the file containing the password of the global administrator used to obtain **`dsreplication status`** information.
- **--outputPath**. Specifies the path (and optionally, the name) for the support data archive file. If the path specifies a filename, the archive is written to that file. If the path specifies a directory, the file is written into that directory with a server-generated name.
- **--useRemoteServer**. Invokes the tool against a remote server instance and streams the output and resulting support data archive back to the client. This option can be especially useful when the server instance is running in a container because it might otherwise be difficult to invoke commands or access files in that container. See also: **--proxyToServerAddress** and **--proxyToServerPort**.
- **--proxyToServerAddress** and **--proxyToServerPort**. Use these options with **--useRemoteServer** to indicate that the support data archive should be retrieved from a server that is not directly accessible but can be accessed through a Directory Proxy Server.

Running the collect-support-data tool

Steps

1. Go to the server root directory.
2. Use the **`collect-support-data`** tool. Make sure to include the host, port number, bind DN, and bind password.

```
$ bin/collect-support-data --hostname 127.0.0.1 --port 389 \
  --bindDN "cn=Directory Manager" --bindPassword secret \
  --serverRoot /opt/PingDirectoryProxy --pid 1234
```

3. Email the zip file to your Authorized Support Provider.

Directory Proxy Server troubleshooting tools

The PingDirectoryProxy Server provides a set of tools that can also be used to obtain information for diagnosing and solving problems.

Server version information

If it becomes necessary to contact your authorized support provider, then it will be important to provide precise information about the version of the Directory Proxy Server software that is in use. If the server is running, then this information can be obtained from the "`cn=Version,cn=monitor`" entry. It can also be obtained using the command:

```
$ bin/status --fullVersion
```

This command outputs a number of important pieces of information, including:

- Major, minor, point and patch version numbers for the server.
- Source revision number from which the server was built.
- Build information including build ID with timestamp, OS, user, Java and JVM version for the build.
- Auxiliary software versions: Jetty, JZlib, SNMP4j (SNMP4J, Agent, Agentx), Groovy, LDAP SDK for Java, and the Server SDK.

LDIF connection handler

The Directory Proxy Server provides an LDIF connection handler that provides a way to request operations that do not require any network communication with the server. This can be particularly helpful if a configuration problem or bug in the server has left a connection handler unusable, or if all worker threads are busy processing operations.

The LDIF connection handler is enabled by default and looks for LDIF files to be placed in the `config/auto-process-ldif` directory. This Directory Proxy Server does not exist by default, but if it is created and an LDIF file is placed in it that contains one or more changes to be processed, then those changes will be applied.

Any changes that can be made over LDAP can be applied through the LDIF connection handler. It is primarily intended for administrative operations like updating the server configuration or scheduling tasks, although other types of changes (including changes to data contained in the server) can be processed. As the LDIF file is processed, a new file is written with comments for each change providing information about the result of processing that change.

Embedded profiler

If the Directory Proxy Server appears to be running slowly, then it is helpful to know what operations are being processed in the server. The JVM Stack Trace monitor entry can be used to obtain a point-in-time snapshot of what the server is doing, but in many cases, it might be useful to have information collected over a period of time.

The embedded profiler is configured so that it is always available but is not active by default so that it has no impact on the performance of the running server. Even when it is running, it has a relatively small impact on performance, but it is recommended that it remain inactive when it is not needed. It can be controlled using the **`dsconfig`** tool or the Administrative Console by managing the "Profiler" configuration object in the "Plugin" object type, available at the standard object level. The `profile-action` property for this configuration object can have one of the following values:

- **start** – Indicates that the embedded profiler should start capturing data in the background.
- **stop** – Indicates that the embedded profiler should stop capturing data and write the information that it has collected to a `logs/profile{timestamp}` file.
- **cancel** – Indicates that the embedded profiler should stop capturing data and discard any information that it has collected.

Any profiling data that has been captured can be examined using the **profiler-viewer** tool. This tool can operate in either a text-based mode, in which case it dumps a formatted text representation of the profile data to standard output, or it can be used in a graphical mode that allows the information to be more easily understood.

Invoking the profile viewer in text-based mode

Steps

- Run the **profile-viewer** command and specify the captured log file using the `--fileName` option.

```
$ bin/profile-viewer --fileName logs/profile.20110101000000Z
```

Invoking the profile viewer in GUI mode

Steps

- Run the **profile-viewer** command and specify the captured log file using the `--fileName` option. To invoke GUI mode, add the option `--useGUI`.

```
$ bin/profile-viewer --fileName logs/profile.20110101000000Z --useGUI
```

Troubleshooting resources for Java applications

Because the PingDirectoryProxy Server is written entirely in Java, it is possible to use standard Java debugging and instrumentation tools when troubleshooting problems with the Directory Proxy Server. In many cases, obtaining the full benefit of these tools requires access to the Directory Proxy Server source code. These Java tools should be used under the advisement of your authorized support provider.

Java troubleshooting tools

The Java Development Kit provides a number of very useful tools to obtain information about Java applications and diagnosing problems. These tools are not included with the Java Runtime Environment (JRE), so the full Java Development Environment (JDK) should always be installed and used to run the PingDirectoryProxy Server.

jps

The **jps** tool is a Java-specific version of the UNIX **ps** tool. It can be used to obtain a list of all Java processes currently running and their respective process identifiers. When invoked by a non-root user, it will list only Java processes running as that user. When invoked by a root user, then it lists all Java processes on the system.

This tool can be used to see if the Directory Proxy Server is running and if a process ID has been assigned to it. This process ID can be used in conjunction with other tools to perform further analysis.

This tool can be run without any arguments, but some of the more useful arguments that include:

- **-v** – Includes the arguments passed to the JVM for the processes that are listed.
- **-m** – Includes the arguments passed to the main method for the processes that are listed.
- **-l** – (lowercase L). Include the fully qualified name for the main class rather than only the base class name.

jstack

The **jstack** tool is used to obtain a stack trace of a running Java process, or optionally from a core file generated if the JVM happens to crash. A stack trace can be extremely valuable when trying to debug a problem, because it provides information about all threads running and exactly what each is doing at the point in time that the stack trace was obtained.

Stack traces are helpful when diagnosing problems in which the server appears to be hung or behaving slowly. Java stack traces are generally more helpful than native stack traces, because Java threads can

have user-friendly names (as do the threads used by the PingDirectoryProxy Server), and the frame of the stack trace may include the line number of the source file to which it corresponds. This is useful when diagnosing problems and often allows them to be identified and resolved quickly.

To obtain a stack trace from a running JVM, use the command:

```
jstack {processID}
```

where {processID} is the process ID of the target JVM as returned by the **jps** command. To obtain a stack trace from a core file from a Java process, use the command:

```
jstack {pathToJava} {pathToCore}
```

where `{pathToJava}` is the path to the java command from which the core file was created, and `{pathToCore}` is the path to the core file to examine. In either case, the stack trace is written to standard output and includes the names and call stacks for each of the threads that were active in the JVM.

In many cases, no additional options are necessary. The "`-l`" option can be added to obtain a long listing, which includes additional information about locks owned by the threads. The "`-m`" option can be used to include native frames in the stack trace.

jmap

The **jmap** tool is used to obtain information about the memory consumed by the JVM. It is very similar to the native **pmap** tool provided by many operating systems. As with the **jstack** tool, **jmap** can be invoked against a running Java process by providing the process ID, or against a core file, like:

```
jmap {processID}
jmap {pathToJava} {pathToCore}
```

Some of the additional arguments include:

- **-dump:live,format=b,file=filename** – Dump the live heap data to a file that can be examined by the **jhat** tool
- **-heap** – Provides a summary of the memory used in the Java heap, along with information about the garbage collection algorithm in use.
- **-histo:live** – Provides a count of the number of objects of each type contained in the heap. If the "`:live`" portion is included, then only live objects are included; otherwise, the count include objects that are no longer in use and are garbage collected.

jhat

The **jhat** (Java Heap Analysis Tool) utility provides the ability to analyze the contents of the Java heap. It can be used to analyze a heap dump file, which is generated if the Directory Proxy Server encounters an out of memory error (as a result of the "`-XX:+HeapDumpOnOutOfMemoryError`" JVM option) or from the use of the **jmap** command with the "`-dump`" option.

The **jhat** tool acts as a web server that can be accessed by a browser in order to query the contents of the heap. Several predefined queries are available to help determine the types of objects consuming significant amounts of heap space, and it also provides a custom query language (OQL, the Object Query Language) for performing more advanced types of analysis.

The **jhat** tool can be launched with the path to the heap dump file, like:

```
jhat /path/to/heap.dump
```

This command causes the **jhat** web server to begin listening on port 7000. It can be accessed in a browser at `http://localhost:7000` (or `http://address:7000` from a remote system). An alternate port number can be specified using the "`-port`" option, like:

```
jhat -port 1234 /path/to/heap.dump
```

To issue custom OQL searches, access the web interface using the URL `http://localhost:7000/oql/` (the trailing slash must be provided). Additional information about the OQL syntax may be obtained in the web interface at `http://localhost:7000/oqlhelp/`.

jstat

The **`jstat`** tool is used to obtain a variety of statistical information from the JVM, much like the **`vmstat`** utility that can be used to obtain CPU utilization information from the operating system. The general manner to invoke it is as follows:

```
jstat {type} {processID} {interval}
```

The `{interval}` option specifies the length of time in milliseconds between lines of output. The `{processID}` option specifies the process ID of the JVM used to run the Directory Proxy Server, which can be obtained by running **`jps`** as mentioned previously. The `{type}` option specifies the type of output that should be provided. Some of the most useful types include:

- **-class** – Provides information about class loading and unloading.
- **-compile** – Provides information about the activity of the JIT complex.
- **-printcompilation** – Provides information about JIT method compilation.
- **-gc** – Provides information about the activity of the garbage collector.
- **-gccapacity** – Provides information about memory region capacities.

Java diagnostic information

In addition to the tools listed in the previous section, the JVM can provide additional diagnostic information in response to certain events.

Garbage Collection Diagnostic Information

You can enable the JVM debugging options to track garbage collection data for your system. The options can impact JVM performance, but they provide valuable data to tune your server when troubleshooting garbage collection issues. While the **`jstat`** utility with the `-gc` option can be used to obtain some information about garbage collection activity, there are additional arguments that can be added to the JVM to use when running the server to provide additional detail.

```
-XX:+PrintGCDetails
-XX:+PrintTenuringDistribution
-XX:+PrintGCApplicationConcurrentTime
-XX:+PrintGCApplicationStoppedTime
-XX:+PrintGCDateStamps
```

To run the Directory Proxy Server with these options, edit the `config/java.properties` file and add them to the end of the line that begins with "`bin/start-server.java-args`". After the file has been saved, invoke the following command to make those new arguments take effect the next time the server is started:

```
$ bin/dsjavaproperties
```

JVM crash diagnostic information

If the JVM itself should happen to crash for some reason, then it generates a fatal error log with information about the state of the JVM at the time of the crash. By default, this file is named `hs_err_pid{processID}.log` and is written into the base directory of the Directory Proxy Server installation. This file includes information on the underlying cause of the JVM crash, information about the threads running and Java heap at the time of the crash, the options provided to the JVM, environment variables that were set, and information about the underlying system.

Troubleshooting resources in the operating system

The underlying operating system also provides a significant amount of information that can help diagnose issues that impact the performance and the stability of the Directory Proxy Server. In some cases,

problems with the underlying system can be directly responsible for the issues seen with the Directory Proxy Server, and in others system, tools can help narrow down the cause of the problem.

Identifying problems with the underlying system

If the underlying system itself is experiencing problems, it can adversely impact the function of applications running on it. To look for problems in the underlying system view the system log file (`/var/log/messages` on Linux). Information about faulted or degraded devices or other unusual system conditions are written there.

Monitoring system data using the PingDataMetrics Server

The PingDataMetrics Server provides collection and storage of performance data from your server topology. You can use the System Utilization Monitor with the PingDataMetrics Server to collect information about the host system CPU, disk, and network utilization on any platform except Linux. If you are not using the PingDataMetrics Server, you do not need to use the system utilization monitor. When data is being collected, it periodically forks the process and executes commands.

For more information about using the System Utilization Monitor, refer to the data collection chapter of the PingDataMetrics Server documentation.

Examining CPU utilization

Observing CPU utilization for the Directory Proxy Server process and the system as a whole provides clues as to the nature of the problem.

System-Wide CPU utilization

To investigate CPU consumption of the system as a whole, use the **`vmstat`** command with a time interval in seconds, like:

```
vmstat 5
```

The specific output of this command varies between different operating systems, but it includes the percentage of the time the CPU was spent executing user-space code (user time), the percentage of time spent executing kernel-space code (system time), and the percentage of time not executing any code (idle time).

If the CPUs are spending most of their time executing user-space code, the available processors are being well-utilized. If performance is poor or the server is unresponsive, it can indicate that the Directory Proxy Server is not optimally tuned. If there is a high system time, it can indicate that the system is performing excessive disk and/or network I/O, or in some cases, there can be some other system-wide problem like an interrupt storm. If the system is mostly idle but the Directory Proxy Server is performing poorly or is unresponsive, there can be a resource constraint elsewhere (for example, waiting on disk or memory access, or excessive lock contention), or the JVM can be performing other tasks like stop-the-world garbage collection that cannot be run heavily in parallel.

Per-CPU utilization

To investigate CPU consumption on a per-CPU basis, use the **`mpstat`** command with a time interval in seconds, like:

```
mpstat 5
```

On Linux systems, it might be necessary to add "`-P ALL`" to the command, like:

```
mpstat -P ALL 5
```

Among other things, this shows the percentage of time each CPU has spent in user time, system time, and idle time. If the overall CPU utilization is relatively low but **`mpstat`** reports that one CPU has a much higher utilization than the others, there might be a significant bottleneck within the server or the JVM might be performing certain types of garbage collection which cannot be run in parallel. On the other hand, if

CPU utilization is relatively even across all CPUs, there is likely no such bottleneck and the issue might be elsewhere.

Per-process utilization

To investigate CPU consumption on a per-process basis, use a command such as the **top**Directory Proxy Server is consuming a significant amount of available CPU, it might be interfering with the ability of the Directory Proxy Server to run effectively.

Examining disk utilization

If the underlying system has a very high disk utilization, it can adversely impact Directory Proxy Server performance. It could delay the ability to read or write database files or write log files. It could also raise concerns for server stability if excessive disk I/O inhibits the ability of the cleaner threads to keep the database size under control.

The **iostat** tool may be used to obtain information about the disk activity on the system.

On Linux systems, **iostat** should be invoked with the "`-x`" argument, like:

```
iostat -x 5
```

A number of different types of information will be displayed, but to obtain an initial feel for how busy the underlying disks are, look at the "%util" column on Linux. This field shows the percentage of the time that the underlying disks are actively servicing I/O requests. A system with a high disk utilization likely exhibits poor Directory Proxy Server performance.

If the high disk utilization is on one or more disks that are used to provide swap space for the system, the system might not have enough free memory to process requests. As a result, it might have started swapping blocks of memory that have not been used recently to disk. This can cause very poor server performance. It is important to ensure that the server is configured appropriately to avoid this condition. If this problem occurs on a regular basis, then the server is likely configured to use too much memory. If swapping is not normally a problem but it does arise, then check to see if there are any other processes running, which are consuming a significant amount of memory, and check for other potential causes of significant memory consumption (for example, large files in a **tmpfs** file system).

Examining process details

There are a number of tools provided by the operating system that can help examine a process in detail.

ps

The standard **ps** tool can be used to provide a range of information about a particular process. For example, the command can be used to display the state of the process, the name of the user running the process, its process ID and parent process ID, the priority and nice value, resident and virtual memory sizes, the start time, the execution time, and the process name with arguments:

```
ps -fly -p {processID}
```

Note that for a process with a large number of arguments, the standard **ps** command displays only a limited set of the arguments based on available space in the terminal window.

pstack

The **pstack** command can be used to obtain a native stack trace of all threads in a process. While a native stack trace might not be as user-friendly as a Java stack trace obtained using **jstack**, it includes threads that are not available in a Java stack trace. For example, the command displays those threads used to perform garbage collection and other housekeeping tasks. The general usage for the **pstack** command is:

```
pstack {processID}
```

dbx / gdb

A process debugger provides the ability to examine a process in detail. Like **`pstack`**, a debugger can obtain a stack trace for all threads in the process, but it also provides the ability to examine a process (or core file) in much greater detail, including observing the contents of memory at a specified address and the values of CPU registers in different frames of execution. The GNU debugger **`gdb`** is widely-used on Linux systems.

Note that using a debugger against a live process interrupts that process and suspends its execution until it detaches from the process. In addition, when running against a live process, a debugger has the ability to actually alter the contents of the memory associated with that process, which can have adverse effects. As a result, it is recommended that the use of a process debugger be restricted to core files and only used to examine live processes under the direction of your authorized support provider.

pfiles / lsof

To examine the set of files that a process is using (including special types of files, like sockets), you can use a tool such as **`lsof`** on Linux systems, (

```
lsof -p {processID}
```

)

Tracing process execution

If a process is unresponsive but is consuming a nontrivial amount of CPU time, or if a process is consuming significantly more CPU time than is expected, it might be useful to examine the activity of that process in more detail than can be obtained using a point-in-time snapshot. For example, if a process is performing a significant amount of disk reads and/or writes, it can be useful to see which files are being accessed. Similarly, if a process is consistently exiting abnormally, then beginning tracing for that process just before it exits can help provide additional information that cannot be captured in a core file (and if the process is exiting rather than being terminated for an illegal operation, then no core file may be available).

This can be accomplished using the **`strace`** tool on Linux (

```
strace -f -p {processID}
```

).

Consult the **`strace`** manual page for additional information.

Problems with SSL communication

Enable TLS debugging in the server to troubleshoot SSL communication issues:

```
$ dsconfig create-debug-target \
  --publisher-name "File-Based Debug Logger" \
  --target-name
 com.unboundid.directory.server.extensions.TLSConnectionSecurityProvider \
  --set debug-level:verbose \
  --set include-throwable-cause:true
```

```
$ dsconfig set-log-publisher-prop \
  --publisher-name "File-Based Debug Logger" \
  --set enabled:true \
  --set default-debug-level:disabled
```

In the `java.properties` file, add `-Djavax.net.debug=ssl` to the `start-server` line, and run `bin/dsjavaproperties` to make the option take effect on a scheduled server restart.

Examining network communication

Because the PingDirectoryProxy Server is a network-based application, it can be valuable to observe the network communication that it has with clients. The Directory Proxy Server itself can provide details about

its interaction with clients by enabling debugging for the protocol or data debug categories, but there may be a number of cases in which it is useful to view information at a much lower level. A network sniffer, like the *tcpdump* tool on Linux, can be used to accomplish this.

There are many options that can be used with these tools, and their corresponding manual pages will provide a more thorough explanation of their use. However, to perform basic tracing to show the full details of the packets received for communication on port 389 with remote host 1.2.3.4, the following command can be used on Linux:

```
tcpdump -i {interface} -n -XX -s 0 host 1.2.3.4 and port 389
```

It does not appear that the **tcpdump** tool provides support for LDAP parsing. However, it is possible to write capture data to a file rather than displaying information on the terminal (using "`-w {path}`" with **tcpdump**), so that information can be later analyzed with a graphical tool like Wireshark, which provides the ability to interpret LDAP communication on any port.

Note that enabling network tracing generally requires privileges that are not available to normal users and therefore may require root access.

Common problems and potential solutions

This section describes a number of different types of problems that can occur and common potential causes for them.

General troubleshooting methodology

When a problem is detected, Ping Identity recommends using the following general methodology to isolate the problem:

1. Run the **bin/status** tool or look at the server status in the Administrative Console. The **status** tool provides a summary of the server's current state with key metrics and a list of recent alerts.
2. Look in the server logs. In particular, view the following logs:
   - logs/errors
   - logs/failed-ops
   - logs/expensive-ops
3. Use system commands, such as **vmstat** and **iostat** to determine if the server is bottle-necked on a system resource like CPU or disk throughput.
4. For performance problem (especially intermittent ones like spikes in response time), enabling the **periodic-stats-logger** can help to isolate problems, because it stores important server performance information on a per-second basis. The **periodic-stats-logger** can save the information in a csv-formatted file that can be loaded into a spreadsheet. The information this logger makes available is very configurable. You can create multiple loggers for different types of information or a different frequency of logging (for example, hourly data in addition to per-second data). For more information, see "Profiling Server Performance Using the Periodic Stats Logger".
5. For replication problem, run **dsreplication status** and look at the `logs/replication` file.
6. For more advanced users, run the **collect-support-data** tool on the system, unzip the archive somewhere, and look through the collected information. This is often useful when administrators most familiar with the Ping Identity Platform do not have direct access to the systems where the production servers are running. They can examine the **collect-support-data** archive on a different server. For more information, see Using the Collect Support Data Tool.

ⓘ **Important:** Run the **collect-support-data** tool whenever there is a problem whose cause is not easily identified, so that this information can be passed back to your authorized support provider before corrective action can be taken.

The Server will not run setup

If the **setup** tool does not run properly, some of the most common reasons include the following:

A suitable Java environment is not available

The PingDirectoryProxy Server requires that Java be installed on the system and made available to the server, and it must be installed prior to running **setup**. If the **setup** tool does not detect that a suitable Java environment is available, it will refuse to run.

To ensure that this does not happen, the **setup** tool should be invoked with an explicitly-defined value for the *JAVA_HOME* environment variable that specifies the path to the Java installation that should be used. For example:

```
env JAVA_HOME=/ds/java ./setup
```

If this still does not work for some reason, then it can be that the value specified in the provided *JAVA_HOME* environment variable can be overridden by another environment variable. If that occurs, try the following command, which should override any other environment variables that can be set:

```
env UNBOUNDID_JAVA_HOME="/ds/java" UNBOUNDID_JAVA_BIN="" ./setup
```

Unexpected arguments provided to the JVM

If the **setup** script attempts to launch the java command with an invalid set of Java arguments, it might prevent the JVM from starting. By default, no special options are provided to the JVM when running setup, but this might not be the case if either the *JAVA_ARGS* or *UNBOUNDID_JAVA_ARGS* environment variable is set. If the **setup** tool displays an error message that indicates that the Java environment could not be started with the provided set of arguments, then invoke the following command before trying to re-run **setup**:

```
unset JAVA_ARGS UNBOUNDID_JAVA_ARGS
```

The Server has already been configured or used

The **setup** tool is only intended to provide the initial configuration for the Directory Proxy Server. It refuses to run if it detects that the **setup** tool has already been run, or if an attempt has been made to start the Directory Proxy Server prior to running the **setup** tool. This protects an existing Directory Proxy Server installation from being inadvertently updated in a manner that could harm an existing configuration or data set.

If the Directory Proxy Server has been previously used and if you want to perform a fresh installation, it is recommended that you first remove the existing installation, create a new one and run **setup** in that new installation. However, if you are confident that there is nothing of value in the existing installation (for example, if a previous attempt to run **setup** failed to complete successfully for some reason but it will refuse to run again), the following steps can be used to allow the **setup** program to run:

- Remove the `config/config.ldif` file and replace it with the `config/update/config.ldif.{revision}` file containing the initial configuration.
- If there are any files or subdirectories below the `db` directory, then remove them.
- If a `config/java.properties` file exists, then remove it.
- If a `lib/setup-java-home` script (or file on Microsoft Windows) exists, then remove it.

The Server will not start

If the Directory Proxy Server does not start, then there are a number of potential causes.

The Server or other administrative tool is already running

Only a single instance of the Directory Proxy Server can run at any time from the same installation root. If an instance is already running, then subsequent attempts to start the server will fail. Similarly, some other administrative operations can also prevent the server from being started. In such cases, the attempt to start the server should fail with a message like:

```
The Directory Proxy Server could not acquire an exclusive lock on file
```

```
/ds/PingDirectoryProxy/locks/server.lock: The exclusive lock requested for
 file
/ds/PingDirectoryProxy/locks/ server.lock was not granted, which indicates
that another process already holds a shared or exclusive lock on that
file. This generally means that another instance of this server is already
running
```

If the Directory Proxy Server is not running (and is not in the process of starting up or shutting down) and there are no other tools running that could prevent the server from being started, and the server still believes that it is running, then it is possible that a previously-held lock was not properly released. In that case, you can try removing all of the files in the `locks` directory before attempting to start the server.

If you wish to have multiple instances running at the same time on the same system, then you should create a completely separate installation in another location on the file system.

There is not enough memory available

When the Directory Proxy Server is started, the JVM attempts to allocate all memory that it has been configured to use. If there is not enough free memory available on the system, then the Directory Proxy Server generates an error message that indicates that the server could not be started with the specified set of arguments. Note that it is possible that an invalid option was provided to the JVM (as described below), but if that same set of JVM arguments has already been used successfully to run the server, then it is more likely that the system does not have enough memory available.

There are a number of potential causes for this:

- If the amount of memory in the underlying system has changed (for example, system memory has been removed, or if the Directory Proxy Server is running in a zone or other type of virtualized container and a change has been made to the amount of memory that container will be allowed to use), then the Directory Proxy Server might need to be re-configured to use a smaller amount of memory than had been previously configured.
- Another process running on the system is consuming a significant amount of memory so that there is not enough free memory available to start the server. If this is the case, then either terminate the other process to make more memory available for the Directory Proxy Server, or reconfigure the Directory Proxy Server to reduce the amount of memory that it attempts to use.
- The Directory Proxy Server was just shut down and an attempt was made to immediately restart it. In some cases, if the server is configured to use a significant amount of memory, then it can take a few seconds for all of the memory that had been in use by the server, when it was previously running, to be released back to the operating system. In that case, run the **`vmstat`** command and wait until the amount of free memory stops growing before attempting to restart the server.
- If the system is configured with one or more memory-backed file systems, verify whether any large files might be consuming a significant amount of memory in any of those locations. If so, remove them or relocate them to a disk-based file system.
- For Linux systems only, if a mismatch exists between the huge pages setting for the JVM and the huge pages reserved in the operating system.

If nothing else works and there is still not enough free memory to allow the JVM to start, then as a last resort, try rebooting the system.

An invalid Java Environment or JVM option was used

If an attempt to start the Directory Proxy Server fails with an error message indicating that no valid Java environment could be found, or indicates that the Java environment could not be started with the configured set of options, then you should first ensure that enough memory is available on the system as described above. If there is a sufficient amount of memory available, then other causes for this error can include the following:

- The Java installation that was previously used to run the server no longer exists (for example, an updated Java environment was installed and the old installation was removed). In that case, update the `config/java.properties` file to reference to path to the new Java installation and run the `bin/dsjavaproperties` command to apply that change.

- The Java installation used to run the server has been updated and the server is trying to use the correct Java installation but one or more of the options that had worked with the previous Java version no longer work with the new version. In that case, it is recommended that the server be re-configured to use the previous Java version, so that it can be run while investigating which options should be used with the new installation.
- If an *UNBOUNDID_JAVA_HOME* or *UNBOUNDID_JAVA_BIN* environment variable is set, then its value may override the path to the Java installation used to run the server as defined in the `config/java.properties` file. Similarly, if an *UNBOUNDID_JAVA_ARGS* environment variable is set, then its value might override the arguments provided to the JVM. If this is the case, then explicitly unset the *UNBOUNDID_JAVA_HOME*, *UNBOUNDID_JAVA_BIN*, and *UNBOUNDID_JAVA_ARGS* environment variables before trying to start the server.

Note that any time the `config/java.properties` file is updated, the **bin/dsjavaproperties** tool must be run to apply the new configuration. If a problem with the previous Java configuration prevents the **bin/dsjavaproperties** tool from running properly, then it can be necessary to remove the `lib/set-java-home` script (or `lib\set-java-home.bat` file on Microsoft Windows) and invoke the **bin/dsjavaproperties** tool with an explicitly-defined path to the Java environment, like:

```
env UNBOUNDID_JAVA_HOME=/ds/java bin/dsjavaproperties
```

An invalid command-line option was provided

There are a small number of arguments that are provided when running the **bin/start-server** command, but in most cases, none are required. If one or more command-line arguments were provided for the **bin/start-server** command and any of them is not recognized, then the server provides an error message indicating that an argument was not recognized and displays version information. In that case, correct or remove the invalid argument and try to start the server again.

The Server has an invalid configuration

If a change is made to the Directory Proxy Server configuration using an officially-supported tool like **dsconfig** or the Administrative Console, the server should validate that configuration change before applying it. However, it is possible that a configuration change can appear to be valid at the time that it is applied, but does not work as expected when the server is restarted. Alternately, a change in the underlying system can cause a previously-valid configuration to become invalid.

In most cases involving an invalid configuration, the Directory Proxy Server displays (and writes to the error log) a message that explains the problem, and this can be sufficient to identify the problem and understand what action needs to be taken to correct it. If for some reason the startup failure does not provide enough information to identify the problem with the configuration, then look in the `logs/config-audit.log` file to see what recent configuration changes have been made with the server online, or in the `config/archived-configs` directory to see if there might have been a recent configuration change resulting from a direct change to the configuration file itself that was not made through a supported configuration interface.

If the server does not start as a result of a recent invalid configuration change, then it can be possible to start the server using the configuration that was in place the last time that the server started successfully (for example, the "last known good" configuration). This can be achieved using the `--useLastKnownGoodConfig` option:

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

Note that if it has been a long time since the last time the server was started and a number of configuration changes have been made since that time, then the last known good configuration can be significantly out of date. In such cases, it can be preferable to manually repair the configuration.

If there is no last known good configuration, if the server no longer starts with the last known good configuration, or if the last known good configuration is significantly out of date, then manually update the configuration by editing the `config/config.ldif` file. In that case, you should make sure that the server is offline and that you have made a copy of the existing configuration before beginning. You might wish

to discuss the change with your authorized support representative before applying it to ensure that you understand the correct change that needs to be made.

ⓘ **Note:** In addition to manually-editing the config file, you can look at previous achived configurations to see if the most recent one works. You can also use the **`ldif-diff`** tool to compare the configurations in the archive to the current configuration to see what is different.

You do not have sufficient permissions

The Directory Proxy Server should only be started by the user or role used to initially install the server. In most cases, if an attempt is made to start the server as a user or role other than the one used to create the initial configuration, then the server will fail to start, because the user will not have sufficient permissions to access files owned by the other user, such as database and log files. However, if the server was initially installed as a non-root user and then the server is started by the root account, then it can no longer be possible to start the server as a non-root user because new files that are created would be owned by root and could not be written by other users.

If the server was inadvertently started by root when it is intended to be run by a non-root user, or if you wish to change the user account that should be used to run the server, then it should be sufficient to simply change ownership on all files in the Directory Proxy Server installation, so that they are owned by the user or role under which the server should run. For example, if the Directory Proxy Server should be run as the "ds" user in the "other" group, then the following command can be used to accomplish this (invoked by the root user):

```
chown -R ds:other /ds/PingDirectoryProxy
```

The Server has crashed or shut itself down

You can first check the current server state by using the **`bin/server-state`** command. If the Directory Proxy Server was previously running but is no longer active, then the potential reasons include the following:

- The Directory Proxy Server was shut down by an administrator. Unless the server was forcefully terminated (for example, using “kill -9”), then messages are written to the `error` and `server.out` logs explaining the reason for the shutdown.
- The Directory Proxy Server was shut down when the underlying system crashed or was rebooted. If this is the case, then running the **`uptime`** command on the underlying system shows that it was recently booted.
- The Directory Proxy Server process was terminated by the underlying operating system for some reason (for example, the out of memory killer on Linux). If this happens, then a message will be written to the system error log.
- The Directory Proxy Server decided to shut itself down in response to a serious problem that had arisen. At present, this should only occur if the server has detected that the amount of usable disk space has become critically low, or if significant errors have been encountered during processing that left the server without any remaining worker threads to process operations. If this happens, then messages are written to the `error` and `server.out` logs (if disk space is available) to provide the reason for the shutdown.
- The JVM in which the Directory Proxy Server was running crashed. If this happens, then the JVM should dump a fatal error log (a `hs_err_pid{processID}.log` file) and potentially a core file.

In the event that the operating system itself crashed or terminated the process, then you should work with your operating system vendor to diagnose the underlying problem. If the JVM crashed or the server shut itself down for a reason that is not clear, then contact your authorized support provider for further assistance.

Conditions for automatic server shutdown

All PingDirectoryProxy Server servers will shutdown in an out of memory condition, a low disk space error state, or for running out of file descriptors. The Directory Server will enter lockdown mode on unrecoverable database environment errors, but can be configured to shutdown instead with this setting:

```
$ dsconfig set-global-configuration-prop \
            --set unrecoverable-database-error-mode:initiate-server-shutdown
```

The Server will not accept client connections

You can first check the current server state by using the **`bin/server-state`** command. If the Directory Proxy Server does not appear to be accepting connections from clients, then potential reasons include the following:

- The Directory Proxy Server is not running.
- The underlying system on which the Directory Proxy Server is installed is not running.
- The Directory Proxy Server is running but is not reachable as a result of a network or firewall configuration problem. If that is the case, then connection attempts should time out rather than be rejected.
- If the Directory Proxy Server is configured to allow secure communication via SSL or StartTLS, then a problem with the key manager and/or trust manager configuration can cause connections to be rejected. If that is the case, then messages should be written to the server access log for each failed connection attempt.
- If the Directory Proxy Server has been configured with a maximum allowed number of connections, then it can be that the maximum number of allowed client connections are already established. If that is the case, then messages should be written to the server access log for each rejected connection attempt.
- If the Directory Proxy Server is configured to restrict access based on the address of the client, then messages should be written to the server access log for each rejected connection attempt.
- If a connection handler encounters a significant error, then it can stop listening for new requests. If this occurs, then a message should be written to the server error log with information about the problem. Another solution is to restart the server. A third option is to restart the connection handler using the LDIF connection handler to make it available again. To do this, create an LDIF file that disables and then re-enables the connection handler, create the `config/auto-process-ldif` directory if it does not already exist, and then copy the LDIF file into it.

The Server is unresponsive

You can first check the current server state by using the **`bin/server-state`** command. If the Directory Proxy Server process is running and appears to be accepting connections but does not respond to requests received on those connections, then potential reasons for this behavior include:

- If all worker threads are busy processing other client requests, then new requests that arrive will be forced to wait in the work queue until a worker thread becomes available. If this is the case, then a stack trace obtained using the **`jstack`** command shows that all of the worker threads are busy and none of them are waiting for new requests to process.

  A dedicated thread pool can be used for processing administrative operations. This thread pool enables diagnosis and corrective action if all other worker threads are processing operations. To request that operations use the administrative thread pool, using the **`ldapsearch`** command for example, use the **`--useAdministrativeSession`** option. The requester must have the **`use-admin-session`** privilege (included for root users). By default, eight threads are available for this purpose. This can be changed with the **`num-administrative-session-worker-threads`** property in the work queue configuration.

> ⓘ **Note:** If all of the worker threads are tied up processing the same operation for a long time, the server will also issue an alert that it might be deadlocked, which may not actually be the case. All threads might be tied up processing unindexed searches.

- If a request handler is stuck performing some expensive processing for a client connection, then other requests sent to the server on connections associated with that request handler is forced to wait until the request handler is able to read data on those connections. If this is the case, then only some of the connections can experience this behavior (unless there is only a single request handler, in which it will impact all connections), and stack traces obtained using the **`jstack`** command shows that a request handler thread is continuously blocked rather than waiting for new requests to arrive. Note that this scenario is a theoretical problem and one that has not appeared in production.
- If the JVM in which the Directory Proxy Server is running is not properly configured, then it can be forced to spend a significant length of time performing garbage collection, and in severe cases, could cause significant interruptions in the execution of Java code. In such cases, a stack trace obtained from a **`pstack`** of the native process should show that most threads are idle but at least one thread performing garbage collection is active. It is also likely that one or a small number of CPUs is 100% busy while all other CPUs are mostly idle. The server will also issue an alert after detecting a long JVM pause (due to garbage collection). The alert will include details of the pause.
- If the JVM in which the Directory Proxy Server is running has hung for some reason, then the **`pstack`** utility should show that one or more threads are blocked and unable to make progress. In such cases, the system CPUs should be mostly idle.
- If a network or firewall configuration problem arises, then attempts to communicate with the server cannot be received by the server. In that case, a network sniffer like **`snoop`** or **`tcpdump`** should show that packets sent to the system on which the Directory Proxy Server is running are not receiving TCP acknowledgement.
- If the system on which the Directory Proxy Server is running has become hung or lost power with a graceful shutdown, then the behavior is often similar to that of a network or firewall configuration problem.

If it appears that the problem is with the Directory Proxy Server software or the JVM in which it is running, then you need to work with your authorized support provider to fully diagnose the problem and determine the best course of action to correct it.

The Server is slow to respond to client requests

If the Directory Proxy Server is running and does respond to clients, but clients take a long time to receive responses, then the problem can be attributable to a number of potential problems. In these cases, use the Periodic Stats Logger, which is a valuable tool to get per-second monitoring information on the Directory Proxy Server. The Periodic Stats Logger can save the information in csv format for easy viewing in a spreadsheet. For more information, see "Profiling Server Performance Using the Periodic Stats Logger". The potential problems that cause slow responses to client requests are as follows:

- The server is not optimally configured for the type of requests being processed, or clients are requesting inefficient operations. If this is the case, then the access log should show that operations are taking a long time to complete and they will likely be unindexed. In that case, updating the server configuration to better suit the requests, or altering the requests to make them more efficient, could help alleviate the problem. In this case, view the expensive operations access log in `logs/expensive-ops`, which by default logs operations that take longer than 1 second. You can also run the **`bin/status`** command or view the status in the Administrative Console to see the Directory Proxy Server's Work Queue information (also see the next bullet point).
- The server is overwhelmed with client requests and has amassed a large backlog of requests in the work queue. This can be the result of a configuration problem (for example, too few worker thread configured), or it can be necessary to provision more systems on which to run the Directory Proxy Server software. Symptoms of this problem appear similar to those experienced when the server is asked to process inefficient requests, but looking at the details of the requests in the access log show that they are not necessarily inefficient requests. Run the **`bin/status`** command to view the Work Queue information. If everything is performing well, you should not see a large queue size or a server that is near 100% busy. The %Busy statistic is calculated as the percentage of worker threads that are busy processing operations.

```
           --- Work Queue ---
           : Recent : Average : Maximum
```

```
-----------:--------:---------:--------
Queue Size : 10  : 1       : 10
% Busy     : 17     : 14      : 100
```

You can also view the expensive operations access log in `logs/expensive-ops`, which by default logs operations that take longer than 1 second.

- The server is not configured to fully cache all of the data in the server, or the cache is not yet primed. In this case, **`iostat`** reports a very high disk utilization. This can be resolved by configuring the server to fully cache all data, and to load database contents into memory on startup. If the underlying system does not have enough memory to fully cache the entire data set, then it might not be possible to achieve optimal performance for operations that need data which is not contained in the cache. For more information, see Disk-Bound Deployments.
- If the JVM is not properly configured, then it will need to perform frequent garbage collection and periodically pause execution of the Java code that it is running. In that case, the server error log should report that the server has detected a number of pauses and can include tuning recommendations to help alleviate the problem.
- If the Directory Proxy Server is configured to use a large percentage of the memory in the system, then it is possible that the system has gotten low on available memory and has begun swapping. In this case, **`iostat`** should report very high utilization for disks used to hold swap space, and commands like **`cat /proc/meminfo`** on Linux can report a large amount of swap memory in use. Another cause of swapping is if swappiness is not set to 0 on Linux. For more information, see Disable File System Swapping.
- If another process on the system is consuming a significant amount of CPU time, then it can adversely impact the ability of the Directory Proxy Server to process requests efficiently. Isolating the processes (for example, using processor sets) or separating them onto different systems can help eliminate this problem.

The Server returns error responses to client requests

If a large number of client requests are receiving error responses, then view the `logs/failed-ops` log, which is an access log for only failed operations. The potential reasons for the error responses include the following:

- If clients are requesting operations that legitimately should fail (for example, they are targeting entries that do not exist, are attempting to update entries in a way that would violate the server schema, or are performing some other type of inappropriate operation), then the problem is likely with the client and not the server.
- If a portion of the Directory Proxy Server data is unavailable (for example, because an online LDIF import or restore is in progress), then operations targeting that data will fail. Those problems will be resolved when the backend containing that data is brought back online. During the outage, it might be desirable to update proxies or load balancers or both to route requests away from the affected server. As of Directory Proxy Server version 3.1 or later, the Directory Proxy Server will indicate that it is in a degraded status and the Directory Proxy Server will route around it.
- If the Directory Proxy Server work queue is configured with a maximum capacity and that capacity has been reached, then the server begins rejecting all new requests until space is available in the work queue. In this case, it might be necessary to alter the server configuration or the client requests or both, so that they can be processed more efficiently, or it might be necessary to add additional server instances to handle some of the workload.
- If an internal error occurs within the server while processing a client request, then the server terminates the connection to the client and logs a message about the problem that occurred. This should not happen under normal circumstances, so you will need to work with your authorized support provider to diagnose and correct the problem.
- If a problem is encountered while interacting with the underlying database (for example, an attempt to read from or write to disk failed because of a disk problem or lack of available disk space), then it can begin returning errors for all attempts to interact with the database until the backend is closed and re-opened and the database has been given a change to recover itself. In these cases, the `je.info.*` file in the database directory should provide information about the nature of the problem.

The Server must disconnect a client connection

If a client connection must be disconnected due to the expense of the client's request, such as an unindexed search across a very large database, perform the following:

- Find the client's connection ID by looking in the **cn=Active Operations,cn=monitor monitor** entry.

```
$ bin/ldapsearch -baseDN cn=monitor "cn=active operations" \
  --bindDN "cn=directory manager"  \
  --bindPassword password
```

- The monitor entry will contain attribute values for **operation-in-progress**, which look like an access log message. Look for the value of **conn** in the client request that should be disconnected. In the following example, the client to be disconnected is requesting a search for **(description=expensive)**, which is on connection 6.

```
dn: cn=Active Operations,cn=monitor
objectClass: top
objectClass: ds-monitor-entry
objectClass: ds-active-operations-monitor-entry
objectClass: extensibleObject
cn: Active Operations
num-operations-in-progress: 2
operation-in-progress: [15/Dec/2014:10:55:35 -0600] SEARCH conn=6 op=3
 msgID=4
     clientIP="10.8.4.21" authDN="cn=app1,ou=applications,dc=example,dc=com"
 base="dc
     =example,dc=com" scope=wholeSubtree filter="(description=expensive)"
 attrs="A
     LL" unindexed=true
operation-in-progress: [15/Dec/2014:10:56:11 -0600] SEARCH conn=7 op=1
 msgID=2
     clientIP="127.0.0.1" authDN="cn=Directory Manager,cn=Root
 DNs,cn=config" base="c
     n=monitor" scope=wholeSubtree filter="(cn=active operations)"
 attrs="ALL"
     num-persistent-searches-in-progress: 0
```

- With the connection ID value, create a file with the following contents, named **disconnect6.ldif**.

```
dn: ds-task-id=disconnect6,cn=scheduled Tasks,cn=tasks
objectClass: top
objectClass: ds-task
objectClass: ds-task-disconnect
ds-task-disconnect-connection-id: 6
ds-task-id: disconnect6
ds-task-class-name:
 com.unboundid.directory.server.tasks.DisconnectClientTask
```

- This LDIF file represents a task entry. The connection ID value 6 is assigned to **ds-task-disconnect-connection-id**. The value for **ds-task-id** value does not follow a specific convention. It must be unique among other task entries currently cached by the server.
- Disconnect the client and cancel the associated operation by adding the task entry to the server:

```
$ bin/ldapmodify --filename disconnect6.ldif  \
  --defaultAdd --bindDN "cn=directory manager" \
  --bindPassword password
```

Problems with the Administrative Console

If a problem arises when trying to use the Administrative Console, then potential reasons for the problem may include the following:

- The web application container used to host the console is not running. If an error occurs while trying to start it, then consult the logs for the web application container.
- If a problem occurs while trying to authenticate to the web application container, then make sure that the target Directory Proxy Server is online. If it is online, then the access log may provide information about the reasons for the authentication failure.
- If a problem occurs while attempting to interact with the Directory Proxy Server instance using the Administrative Console, then the access and error logs for that Directory Proxy Server instance might provide additional information about the underlying problem.

Problems with the Administrative Console: JVM memory issues

Console runs out of memory (PermGen). An inadequate PermSize setting in the server, while hosting web applications like the Administrative Console may result in errors like this in the error log:

```
[02/Mar/2016:07:50:27.017 -0600] threadID=2 category=UTIL
    severity=SEVERE_ERROR msgID=-1 msg="The server experienced an unexpected
    error. Please report this problem and include this log file.
    OutOfMemoryError: PermGen space
    ()\ncom.unboundid.directory.server.core.DirectoryServer.uncaughtException
    (DirectoryServer.java:15783)\njava.lang.ThreadGroup.uncaughtException
    (ThreadGroup.java:1057)\njava.lang.ThreadGroup.uncaughtException
    (ThreadGroup.java:1052)\njava.lang.ThreadGroup.uncaughtException
    (ThreadGroup.java:1052)\njava.lang.Thread.dispatchUncaughtException
    (Thread.java:1986)\nBuild revision: 22496\n"
```

This is only relevant for servers running Java 7.

Global Index Growing Too Large

If the global index appears to be growing too large, you can reload from the backend directory servers. Use the **`reload-index`** tool with the `--fromDS` option, overriding the configuration of the `prime-index-source` property. You can do this on a one off basis if the global index appears to be growing too large as follows:

```
$ bin/reload-index \
  --bindPassword password \
  --baseDN "dc=example,dc=com" \
  --fromDS
```

Forgotten Proxy User Password

If you have forgotten the password you set for the `cn=Proxy User` entry, you can work around the problem as follows:

- You can temporarily add a second password to the proxy user entry so that you can transition all of the proxy server instances to the new password. However, you should have multiple passwords on the `cn=Proxy User` entry for the shortest time possible.
- If you do not know the clear-text value, then you can use the encrypted value when configuring the new Directory Proxy Server. The encryption scheme allows reversible passwords that are stored in the server configuration so that they can be decrypted by any server instance.
- You can create a new root user in the directory server instances with the appropriate set of privileges and have the new proxy server instance use that account to authenticate. Since it is not a good idea to have an account for which you do not know the password, you may want to update all of the other proxy server instances to use the new account.
- You can use a protocol analyzer like **`snoop`** or **`Wireshark`**, to capture the password from the network communication.

Providing information for support cases

If a problem arises that you are unable to fully diagnose and correct on your own, then contact your authorized support provider for assistance. To ensure that the problem can be addressed as quickly as possible, be sure to provide all of the information that the support personnel may need to fully understand

the underlying cause by running the **`collect-support-data`** tool, and then sending the generated zip file to your authorized support provider. It is good practice to run this tool and send the ZIP file to your authorized support provider before any corrective action has taken place.

SCIM 1.1 and 2.0 servlet extensions management

The PingDirectoryProxy Server provides two System for Cross-domain Identity Management (SCIM) servlet extensions to facilitate moving users to, from, and between cloud-based Software-as-a-Service (SaaS) applications in a secure, fast, and simple way. SCIM is an alternative to LDAP, allowing identity data provisioning between cloud-based applications over HTTPS. One servelt implements SCIM 1.1 and the other servlet implements SCIM 2.

This section describes fundamental SCIM concepts and provides information on configuring SCIM on your server.

Overview of SCIM 1.1 fundamentals

Understanding the basic concepts of SCIM can help you use the SCIM extension to meet your deployment needs. SCIM allows you to:

- **Provision identities.** Through the API, you have access to the basic create, read, update, and delete functions, as well as other special functions.
- **Provision groups.** SCIM also allows you to manage groups.
- **Interoperate using a common schema.** SCIM provides a well-defined, platform-neutral user and group schema, as well as a simple mechanism to extend it.

The SCIM extension implements the 1.1 version of the SCIM specification. Familiarize yourself with this specification to help you understand and make efficient use of the SCIM extension and the SCIM SDK. The SCIM specifications are located on the Simplecloud website.

ⓘ **Note:** SCIM 1.1 will be deprecated in a future release.

Summary of SCIM 1.1 protocol support

PingDirectoryProxy Server supports all required features of the SCIM 1.1 protocol and most optional features. The following table describes SCIM features and whether they are supported.

SCIM Protocol Support

SCIM Feature	Supported
Etags	Yes
JSON	Yes
XML*	Yes
Authentication/Authorization	Yes, via HTTP basic authentication or OAuth 2.0 bearer tokens
Service Provider Configuration	Yes
Schema	Yes
User resources	Yes
Group resources	Yes
User-defined resources	Yes
Resource retrieval via GET	Yes

SCIM Feature	Supported
List/query resources	Yes
Query filtering*	Yes
Query result sorting*	Yes
Query result pagination*	Yes (Directory Server, not Directory Proxy Server)
Resource updates via PUT	Yes
Partial resource updates via PATCH*	Yes
Resource deletes via DELETE	Yes
Resource versioning*	Yes (requires configuration for updated servers)
Bulk*	Yes
HTTP method overloading	Yes
Raw LDAP Endpoints**	Yes

* denotes an optional feature of the SCIM protocol.

** denotes a PingDirectoryProxy Server extension to the basic SCIM functionality.

About the Identity Access API

The PingDirectory Server, PingDirectoryProxy Server, and PingDataSync Server support an extension to the SCIM 1.1 standard called the Identity Access API. The Identity Access API provides an alternative to LDAP by supporting CRUD (create, read, update, and delete) operations to access directory server data over an HTTP connection.

SCIM 1.1 and the Identity Access API are provided as a unified service through the SCIM HTTP Servlet Extension. The SCIM HTTP Servlet Extension can be configured to only enable core SCIM resources (e.g., 'Users' and 'Groups'), only LDAP object classes (e.g., `top`, `domain`, `inetOrgPerson`, or `groupOfUniqueNames`), or both. Because SCIM and the Identity Access API have different schemas, if both are enabled, there may be two representations with different schemas for any resources defined in the `scim-resources.xml` file: the SCIM representation and the raw LDAP representation. Likewise, because resources are exposed by an LDAP object class, and because these are hierarchical (e.g., `top` --> `person` --> `organizationalPerson` --> `inetOrgPerson`, etc.), a client application can access an entry in multiple ways due to the different paths/URIs to a given resource.

This chapter provides information on configuring the SCIM and the Identity Access API services on the PingDirectory Server.

Creating your own SCIM 1.1 application

The System for Cross-domain Identity Management (SCIM) is an open initiative designed to make moving identity data to, from, and between clouds standard, secure, fast, and easy. The SCIM SDK is a pre-packaged collection of libraries and extensible classes that provides developers with a simple, concrete API to interact with a SCIM service provider.

The SCIM 1.1 SDK is available for download at https://github.com/pingidentity/scim.

ⓘ **Note:** The value of a read-only SCIM attribute can be set by a POST operation if the SCIM attribute is a custom attribute in the **`scim-resource.xml`** config file, but not if the SCIM attribute is a core SCIM attribute.

Configuring SCIM 1.1

This section discusses details about the PingDirectoryProxy Server implementation of the SCIM protocol. Before reading this chapter, familiarize yourself with the SCIM Protocol specification, available on the Simplecloud website.

Before You Begin

To set up your SCIM servlet extension, the Directory Proxy Server provides a **`dsconfig`** batch file, `scim-config-ds.dsconfig`, that contains the necessary commands to set up a SCIM extension, which is located in the `config` directory. The batch file assumes that the JKS key and trust manager providers are already enabled. Setup examples are presented in later sections.

The SCIM resource mappings are defined by the `scim-resources.xml` file located in the `config` directory. This file defines the SCIM schema and maps it to the LDAP schema. This file can be customized to define and expose deployment specific resources. See *Managing the SCIM Schema* for more information.

Configuring the SCIM 1.1 servlet extension

The Directory Proxy Server provides a default SCIM HTTP Servlet Extension that can be enabled and configured using a **`dsconfig`** batch script located in the `config` directory. The script runs a series of commands that enables an HTTPS Connection Handler, increases the level of detail logged by the HTTP Detailed Access log publisher, and adds access controls to allow access to LDAP controls used by the SCIM Servlet Extension. There are additional optional configurations (e.g., changing the log format, enable `entryDN` virtual attribute and using VLV indexes) that you can make by altering the `dsconfig` batch script.

The SCIM resource mappings are defined by the `scim-resources.xml` file located in the `config` directory. This file defines the SCIM schema and maps it to the LDAP schema. This file can be customized to define and expose deployment specific resources. See *Managing the SCIM Schema* for more information.

The following procedures show how to configure SCIM on the server. The first example procedure shows the steps to manually configure SCIM without using the script. The second example procedure uses the `dsconfig` batch script to configure SCIM.

Configuring the SCIM servlet extension

Steps

1. Before you enable the SCIM servlet extension, add access controls on each of the backend Directory Servers to allow read access to operational attributes used by the SCIM Servlet Extension. We recommend using the following non-interactive command to add access control instructions, rather than its **`dsconfig`** interactive equivalent.

```
$ bin/dsconfig set-access-control-handler-prop \
  --add 'global-aci:(targetattr="entryUUID || entryDN || ds-entry-unique-id
 ||
    createTimestamp || modifyTimestamp")
    (version 3.0;acl "Authenticated read access to operational attributes \
    used by the SCIM servlet extension"; allow (read,search,compare)
    userdn="ldap:///all";)'
```

2. On the Directory Proxy Server, enable the SCIM servlet extension by running the **`dsconfig`** batch file.

```
$ bin/dsconfig --batch-file config/scim-config-proxy.dsconfig
```

3. The **`dsconfig`** batch file must be edited to use the correct request processor name and base DN name(s) for the `set-request-processor-prop` and `set-root-dse-backend-prop` commands, respectively, as described in the "Configuring LDAP Control Support on All Request Processors" and "SCIM Servlet Extension Authentication" sections later in the chapter.

Enabling resource versioning

About this task

Resource versioning is enabled by default in new installations. Upgraded servers that had SCIM enabled need additional configuration to enable resource versioning.

Steps

1. Enable the **`ds-entry-checksum`** virtual attribute.

```
$ bin/dsconfig set-virtual-attribute-prop \
                        --name ds-entry-checksum \
                        --set enabled:true
```

2. Remove any existing access controls required by SCIM for read access to operational attributes:

```
$ bin/dsconfig set-access-control-handler-prop \
                        --remove 'global-aci:(targetattr="entryUUID ||
 entryDN || ds-entry-unique-id || createTimestamp || ds-create-time ||
 modifyTimestamp || ds-update-time")(version 3.0;acl "Authenticated read
 access to operational attributes used by the SCIM servlet extension"; allow
 (read,search,compare) userdn="ldap:///all"'
```

3. On the backend Directory Server, make sure new access controls required by SCIM for read access to operational attributes are enabled with the following command. If this ACI is not present, issues will occur when a SCIM client tries to authenticate with an non-root DN.

```
$ bin/dsconfig set-access-control-handler-prop \
                        --add  'global-aci:(targetattr="entryUUID ||
 entryDN || ds-entry-unique-id || createTimestamp || ds-create-time ||
 modifyTimestamp || ds-update-time || ds-entry-checksum")(version 3.0;acl
 "Authenticated read access to operational attributes used by the SCIM
 servlet extension"; allow (read,search,compare) userdn="ldap:///all"'
```

Configuring LDAP Control Support on All Request Processors (Proxy Only)

You need to configure support for the required LDAP controls on all request processors handling LDAP requests that result from SCIM requests. Change the request processor name that was provided as an example and repeat the command for all additional request processors.

To Configure LDAP Control Support on All Request Processors

Steps

- Use **`dsconfig`** to change the request processor name that was provided as an example and repeat the command for all additional request processors. Make sure to use your deployment's request processor name.

```
$ bin/dsconfig set-request-processor-prop \
  --processor-name dc_example_dc_com-req-processor \
  --add supported-control-oid:1.2.840.113556.1.4.319 \
  --add supported-control-oid:1.2.840.113556.1.4.473 \
  --add supported-control-oid:2.16.840.1.113730.3.4.9
```

SCIM 1.1 servlet extension authentication

The SCIM 1.1 servlet supports authentication using either the HTTP Basic authentication scheme, or OAuth 2.0 bearer tokens. When authenticating using HTTP Basic authentication, the SCIM 1.1 servlet attempts to correlate the user name component of the Authorization header to a DN in the Directory Proxy Server. If the user name value cannot be parsed directly as a DN, it is correlated to a DN using an Identity Mapper. The DN is then used in a simple bind request to verify the password.

In deployments that use an OAuth authorization server, the SCIM 1.1 extension can be configured to authenticate requests using OAuth bearer tokens. The SCIM 1.1 extension supports authentication with OAuth 2.0 bearer tokens (per RFC 6750) using an OAuth Token Handler Server SDK Extension. Because the OAuth 2.0 specification does not specify how contents of a bearer token are formatted, PingDirectoryProxy Server provides the token handler API to decode incoming bearer tokens and extract or correlate associated authorization DNs.

Neither HTTP Basic authentication nor OAuth 2.0 bearer token authentication are secure unless SSL is used to encrypt the HTTP traffic.

Enabling HTTPS communications

If you want the SCIM HTTP connection handler to use SSL, which is mandated by the SCIM specification, you need to enable a Key Manager provider and Trust Manager provider.

To enable SSL during the Directory Proxy Server's initial setup, include the `--ldapsPort` and the `--generateSelfSignedCertificate` arguments with the **`setup`** command. If your server already has a certificate that you would like to use, set the `key-manager-provider` to the value you set when you enabled SSL in the Directory Proxy Server, or define a new key manager provider (see Configuring HTTP Connection Handlers).

Configuring basic authentication using an identity mapper

About this task

By default, the SCIM servlet is configured to use the Exact Match Identity Mapper, which matches against the **`uid`** attribute. In this example, an alternate Identity Mapper is created so that clients can authenticate using **`cn`** values.

Steps

1. Create a new Identity Mapper that uses a match attribute of cn.

```
$ bin/dsconfig create-identity-mapper \
  --mapper-name "CN Identity Mapper" \
  --type exact-match \
  --set enabled:true \
  --set match-attribute:cn
```

2. Configure the SCIM servlet to use the new Identity Mapper.

```
$ bin/dsconfig set-http-servlet-extension-prop \
  --extension-name SCIM \
  --set "identity-mapper:CN Identity Mapper"
```

Enabling OAuth authentication

About this task

To enable OAuth authentication, you need to create an implementation of the `OAuthTokenHandler` using the API provided in the Server SDK. For details on creating an `OAuthTokenHandler` extension, see the Server SDK documentation.

Steps

1. Install your OAuth token handler on the server using **`dsconfig`**.

```
$ bin/dsconfig create-oauth-token-handler \
  --handler-name ExampleOAuthTokenHandler \
  --type third-party \
  --set extension-
class:com.unboundid.directory.sdk.examples.ExampleOAuthTokenHandler
```

2. Configure the SCIM servlet extension to use it as follows:

```
$ bin/dsconfig set-http-servlet-extension-prop \
  --extension-name SCIM \
  --set oauth-token-handler:ExampleOAuthTokenHandler
```

Using HTTP basic authentication with bare UID on the Directory Proxy Server

As discussed above, clients can authenticate to the SCIM extension using HTTP basic authentication and a bare UID value. However, when a SCIM extension is hosted by a Directory Proxy Server, the server needs to be explicitly configured with the names of subordinate base DNs to search. To do this, run the following command on the Directory Proxy Server for every base DN that may be accessed via SCIM. Make sure to specify your deployment's subordinate base DN.

```
$ bin/dsconfig set-root-dse-backend-prop \
  --set subordinate-base-dn:dc=example,dc=com
```

Verifying the SCIM 1.1 servlet extension configuration

About this task

You can verify the configuration of the SCIM 1.1 extension by navigating to a SCIM resource URL via the command line or through a browser window.

To verify the SCIM servlet extension configuration:

Steps

- Run **`curl`** to verify that the SCIM extension is running. The `-k` (or `--insecure`) option is used to turn off curl's verification of the server certificate, since the example Directory Proxy Server is using a self-signed certificate.

```
$ curl -u "cn=Directory Manager:password" \
-k "https://localhost:8443/scim/ServiceProviderConfigs"

{"schemas":["urn:scim:schemas:core:1.0"],"id":"urn:scim:schemas:core:1.0",
"patch":{"supported":true},"bulk":{"supported":true,"maxOperations":10000,
"maxPayloadSize":10485760},"filter":{"supported":true,"maxResults":100},
"changePassword":{"supported":true},"sort":{"supported":true},
"etag":{"supported":false},"authenticationSchemes":[{"name":"HttpBasic",
"description":"The HTTP Basic Access Authentication scheme. This scheme is
not considered to be a secure method of user authentication (unless used in
conjunction with some external secure system such as SSL), as the user
name and password are passed over the network as cleartext.","specUrl":
"http://www.ietf.org/rfc/rfc2617","documentationUrl":
"http://en.wikipedia.org/wiki/Basic_access_authentication"}]}
```

- If the user ID is a valid DN (such as `cn=Directory Manager`), the SCIM extension authenticates by binding to the Directory Proxy Server as that user. If the user ID is not a valid DN, the SCIM extension searches for an entry with that `uid` value, and binds to the server as that user. To verify authentication to the server as the user with the `uid` of `user.0`, run the following command:

```
$ curl -u "user.0:password" \
  -k "https://localhost:8443/scim/ServiceProviderConfigs"
```

Configuring advanced SCIM 1.1 extension features

The following sections show how to configure advanced SCIM 1.1 servlet extension features, such as bulk operation implementation, mapping SCIM resource IDs, and transformations.

Managing the SCIM 1.1 schema

This section describes the SCIM schema and provides information on how to map LDAP schema to the SCIM resource schema.

About the SCIM schema

SCIM provides a common user schema and extension model, making it easier to interoperate with multiple Service Providers. The core SCIM schema defines a concrete schema for user and group resources that encompasses common attributes found in many existing schemas.

Each attribute is defined as either a single attribute, allowing only one instance per resource, or a multi-valued attribute, in which case several instances may be present for each resource. Attributes may be defined as simple, name-value pairs or as complex structures that define sub-attributes.

While the SCIM schema follows an object extension model similar to object classes in LDAP, it does not have an inheritance model. Instead, all extensions are additive, similar to LDAP Auxiliary Object Classes.

Mapping the LDAP schema to the SCIM resource schema

The resources configuration file is an XML file that is used to define the SCIM resource schema and its mapping to LDAP schema. The default configuration of the `scim-resources.xml` file provides definitions for the standard SCIM Users and Groups resources, and mappings to the standard LDAP `inetOrgPerson` and `groupOfUniqueNames` object classes.

The default configuration may be customized by adding extension attributes to the Users and Groups resources, or by adding new extension resources. The resources file is composed of a single `<resources>` element, containing one or more `<resource>` elements.

For any given SCIM resource endpoint, only one `<LDAPAdd>` template can be defined, and only one `<LDAPSearch>` element can be referenced. If entries of the same object class can be located under different subtrees or base DNs of the Directory Proxy Server, then a distinct SCIM resource must be defined for each unique entry location in the Directory Information Tree. This can be implemented in many ways. For example:

- Create multiple SCIM servlets, each with a unique `scim-resources.xml` configuration, and each running under a unique HTTP connection handler.
- Create multiple SCIM servlets, each with a unique `scim-resources.xml` configuration, each running under a single, shared HTTP connection handler, but each with a unique context path.

Note that LDAP attributes are allowed to contain characters that are invalid in XML (because not all valid UTF-8 characters are valid XML characters). The easiest and most-correct way to handle this is to make sure that any attributes that may contain binary data are declared using "dataType=binary" in the `scim-resources.xml` file. Likewise, when using the Identity Access API make sure that the underlying LDAP schema uses the Binary or Octet String attribute syntax for attributes which may contain binary data. This will cause the server to automatically base64-encode the data before returning it to clients and will also make it predictable for clients because they can assume the data will always be base64-encoded.

However, it is still possible that attributes that are not declared as binary in the schema may contain binary data (or just data that is invalid in XML), and the server will always check for this before returning them to the client. If the client has set the content-type to XML, then the server may choose to base64-encode any values which are found to include invalid XML characters. When this is done, a special attribute is added to the XML element to alert the client that the value is base64-encoded. For example:

```
<scim:value base64Encoded="true">AAABPB0EBZc=</scim:value>
```

The remainder of this section describes the mapping elements available in the `scim-resources.xml` file.

About the `resource` element

A `resource` element has the following XML attributes:

- `schema`: a required attribute specifying the SCIM schema URN for the resource. Standard SCIM resources already have URNs assigned for them, such as `urn:scim:schemas:core:1.0`. A new URN must be obtained for custom resources using any of the standard URN assignment methods.
- `name`: a required attribute specifying the name of the resource used to access it through the SCIM REST API.
- `mapping`: a custom Java class that provides the logic for the resource mapper. This class must extend the `com.unboundid.scim.ldap.ResourceMapper` class.

A resource element contains the following XML elements in sequence:

- `description`: a required element describing the resource.
- `endpoint`: a required element specifying the endpoint to access the resource using the SCIM REST API.
- `LDAPSearchRef`: a mandatory element that points to an `LDAPSearch` element. The `LDAPSearch` element allows a SCIM query for the resource to be handled by an LDAP service and also specifies how the SCIM resource ID is mapped to the LDAP server.
- `LDAPAdd`: an optional element specifying information to allow a new SCIM resource to be added through an LDAP service. If the element is not provided then new resources cannot be created through the SCIM service.
- `attribute`: one or more elements specifying the SCIM attributes for the resource.

About the `attribute` element

An `attribute` element has the following XML attributes:

- `schema`: a required attribute specifying the schema URN for the SCIM attribute. If omitted, the schema URN is assumed to be the same as that of the enclosing resource, so this only needs to be provided for SCIM extension attributes. Standard SCIM attributes already have URNs assigned for them, such as urn:scim:schemas:core:1.0. A new URN must be obtained for custom SCIM attributes using any of the standard URN assignment methods.
- `name`: a required attribute specifying the name of the SCIM attribute.
- `readOnly`: an optional attribute indicating whether the SCIM sub-attribute is not allowed to be updated by the SCIM service consumer. The default value is false.
- `required`: an optional attribute indicating whether the SCIM attribute is required to be present in the resource. The default value is false.

An attribute element contains the following XML elements in sequence:

- `description`: a required element describing the attribute. Then just one of the following elements:
  - `simple`: specifies a simple, singular SCIM attribute.
  - `complex`: specifies a complex, singular SCIM attribute.
  - `simpleMultiValued`: specifies a simple, multi-valued SCIM attribute.
  - `complexMultiValued`: specifies a complex, multi-valued SCIM attribute.

About the `simple` element

A `simple` element has the following XML attributes:

- `dataType`: a required attribute specifying the simple data type for the SCIM attribute. The following values are permitted: binary, boolean, dateTime, decimal, integer, string.
- `caseExact`: an optional attribute that is only applicable for string data types. It indicates whether comparisons between two string values use a case-exact match or a case-ignore match. The default value is false.

A simple element contains the following XML element:

- `mapping`: an optional element specifying a mapping between the SCIM attribute and an LDAP attribute. If this element is omitted, then the SCIM attribute has no mapping and the SCIM service ignores any values provided for the SCIM attribute.

About the `complex` element

The `complex` element does not have any XML attributes. It contains the following XML element:

- `subAttribute`: one or more elements specifying the sub-attributes of the complex SCIM attribute, and an optional mapping to LDAP. The standard `type`, `primary`, and `display` sub-attributes do not need to be specified.

About the `simpleMultivalued` element

A `simpleMultiValued` element has the following XML attributes:

- `childName`: a required attribute specifying the name of the tag that is used to encode values of the SCIM attribute in XML in the REST API protocol. For example, the tag for the standard emails SCIM attribute is email.
- `dataType`: a required attribute specifying the simple data type for the plural SCIM attribute (i.e. the data type for the value sub-attribute). The following values are permitted: `binary`, `boolean`, `dateTime`, `integer`, `string`.
- `caseExact`: an optional attribute that is only applicable for string data types. It indicates whether comparisons between two string values use a case-exact match or a case-ignore match. The default value is false.

A simpleMultiValued element contains the following XML elements in sequence:

- `canonicalValue`: specifies the values of the type sub-attribute that is used to label each individual value, and an optional mapping to LDAP.
- `mapping`: an optional element specifying a default mapping between the SCIM attribute and an LDAP attribute.

About the `complexMultiValued` element

A `complexMultiValued` element has the following XML attribute:

- `tag`: a required attribute specifying the name of the tag that is used to encode values of the SCIM attribute in XML in the REST API protocol. For example, the tag for the standard addresses SCIM attribute is address.

A `complexMultiValued` element contains the following XML elements in sequence:

- `subAttribute`: one or more elements specifying the sub-attributes of the complex SCIM attribute. The standard type, primary, and display sub-attributes do not need to be specified.
- `canonicalValue`: specifies the values of the type sub-attribute that is used to label each individual value, and an optional mapping to LDAP.

About the `subAttribute` element

A `subAttribute` element has the following XML attributes:

- `name`: a required element specifying the name of the sub-attribute.
- `readOnly`: an optional attribute indicating whether the SCIM sub-attribute is not allowed to be updated by the SCIM service consumer. The default value is false.
- `required`: an optional attribute indicating whether the SCIM sub-attribute is required to be present in the SCIM attribute. The default value is false.
- `dataType`: a required attribute specifying the simple data type for the SCIM sub-attribute. The following values are permitted: binary, boolean, dateTime, integer, string.
- `caseExact`: an optional attribute that is only applicable for string data types. It indicates whether comparisons between two string values use a case-exact match or a case-ignore match. The default value is false.

A `subAttribute` element contains the following XML elements in sequence:

- `description`: a required element describing the sub-attribute.

- `mapping`: an optional element specifying a mapping between the SCIM sub-attribute and an LDAP attribute. This element is not applicable within the complexMultiValued element.

About the `canonicalValue` element

A `canonicalValue` element has the following XML attribute:

- `name`: specifies the value of the type sub-attribute. For example, work is the value for emails, phone numbers and addresses intended for business purposes.

A `canonicalValue` element contains the following XML element:

- `subMapping`: an optional element specifying mappings for one or more of the sub-attributes. Any sub-attributes that have no mappings will be ignored by the mapping service.

About the `mapping` element

A `mapping` element has the following XML attributes:

- `ldapAttribute`: A required element specifying the name of the LDAP attribute to which the SCIM attribute or sub-attribute map.
- `transform`: An optional element specifying a transformation to apply when mapping an attribute value from SCIM to LDAP and vice-versa. The available transformations are described in the *Mapping LDAP Entries to SCIM Using the SCIM-LDAP API* section.

About the `subMapping` element

A `subMapping` element has the following XML attributes:

- `name`: a required element specifying the name of the sub-attribute that is mapped.
- `ldapAttribute`: a required element specifying the name of the LDAP attribute to which the SCIM sub-attribute maps.
- `transform`: an optional element specifying a transformation to apply when mapping an attribute value from SCIM to LDAP and vice-versa. The available transformations are described later. The available transformations are described in *Mapping LDAP Entries to SCIM Using the SCIM-LDAP API*.

About the `LDAPSearch` element

An `LDAPSearch` element contains the following XML elements in sequence:

- `baseDN`: a required element specifying one or more LDAP search base DNs to be used when querying for the SCIM resource.
- `filter`: a required element specifying an LDAP filter that matches entries representing the SCIM resource. This filter is typically an equality filter on the LDAP object class.
- `resourceIDMapping`: an optional element specifying a mapping from the SCIM resource ID to an LDAP attribute. When the element is omitted, the resource ID maps to the LDAP entry DN. Note The `LDAPSearch` element can be added as a top-level element outside of any <Resource> elements, and then referenced within them via an ID attribute.

> ⓘ **Note:** The `LDAPSearch` element can be added as a top-level element outside of any <Resource> elements, and then referenced within them via an ID attribute.

About the `resourceIDMapping` element

The `resourceIDMapping` element has the following XML attributes:

- `ldapAttribute`: a required element specifying the name of the LDAP attribute to which the SCIM resource ID maps.
- `createdBy`: a required element specifying the source of the resource ID value when a new resource is created by the SCIM consumer using a POST operation. Allowable values for this element include `scim-consumer`, meaning that a value must be present in the initial resource content provided by the SCIM consumer, or Directory Proxy Server, meaning that a value is automatically provided by the Directory Proxy Server (as would be the case if the mapped LDAP attribute is entryUUID).

The following example illustrates an `LDAPSearch` element that contains a `resourceIDMapping` element:

```
<LDAPSearch id="userSearchParams">
  <baseDN>ou=people,dc=example,dc=com</baseDN>
  <filter>(objectClass=inetOrgPerson)</filter>
  <resourceIDMapping ldapAttribute="entryUUID" createdBy="directory"/>
</LDAPSearch>
```

About the `LDAPAdd` element

An LDAPAdd element contains the following XML elements in sequence:

- `DNTemplate`: a required element specifying a template that is used to construct the DN of an entry representing a SCIM resource when it is created. The template may reference values of the entry after it has been mapped using `{ldapAttr}`, where `ldapAttr` is the name of an LDAP attribute.
- `fixedAttribute`: zero or more elements specifying fixed LDAP values to be inserted into the entry after it has been mapped from the SCIM resource.

About the `fixedAttribute` element

A `fixedAttribute` element has the following XML attributes:

- `ldapAttribute`: a required attribute specifying the name of the LDAP attribute for the fixed values.
- `onConflict`: an optional attribute specifying the behavior when the LDAP entry already contains the specified LDAP attribute. The value merge indicates that the fixed values should be merged with the existing values. The value overwrite indicates that the existing values are to be overwritten by the fixed values. The value preserve indicates that no changes should be made. The default value is merge.

A `fixedAttribute` element contains one or more `fixedValue` XML element, which specify the fixed LDAP values.

Validating the updated SCIM schema

The PingDirectoryProxy Server SCIM extension is bundled with an XML Schema document, `resources.xsd`, which describes the structure of a `scim-resources.xml` resource configuration file. After updating the resource configuration file, you should confirm that its contents are well-formed and valid using a tool such as `xmllint`.

For example, you could validate your updated file as follows:

```
$ xmllint --noout --schema resources.xsd scim-resources.xml
scim-resources.xml validates
```

Mapping SCIM resource IDs

The default `scim-resources.xml` configuration maps the SCIM resource ID to the LDAP `entryUUID` attribute. The `entryUUID` attribute, whose read-only value is assigned by the Directory Proxy Server, meets the requirements of the SCIM specification regarding resource ID immutability. However, configuring a mapping to the attribute may result in inefficient group processing, since LDAP groups use the entry DN as the basis of group membership. The resource configuration allows the SCIM resource ID to be mapped to the LDAP entry DN. However, the entry DN does not meet the requirements of the SCIM specification regarding resource ID immutability. LDAP permits entries to be renamed or moved, thus modifying the DN. Likewise, you can use the Identity Access API to change the value of an entry's RDN attribute, thereby triggering a MODDN operation.

A resource may also be configured such that its SCIM resource ID is provided by an arbitrary attribute in the request body during POST operations. This SCIM attribute must be mapped to an LDAP attribute so that the SCIM resource ID may be stored in the Directory Proxy Server. By default, it is the responsibility of the SCIM client to guarantee ID uniqueness. However, the UID Unique Attribute Plugin may be used by the Directory Proxy Server to enforce attribute value uniqueness. For information about the UID Unique Attribute Plugin, see "Working with the UID Unique Attribute plugin" in the *PingDirectory Server Administration Guide*.

ⓘ **Note:** Resource IDs may not be mapped to virtual attributes. For more information about configuring SCIM Resource IDs, see "About the `<resourceIDMapping>` Element".

Using pre-defined transformations

Transformations are required to change SCIM data types to LDAP syntax values. The following pre-defined transformations may be referenced by the transform XML attribute:

- `com.unboundid.scim.ldap.BooleanTransformation`. Transforms SCIM boolean data type values to LDAP Boolean syntax values and vice-versa.
- `com.unboundid.scim.ldap.GeneralizedTimeTransformation`. Transforms SCIM dateTime data type values to LDAP Generalized Time syntax values and vice-versa.
- `com.unboundid.scim.ldap.PostalAddressTransformation`. Transforms SCIM formatted address values to LDAP Postal Address syntax values and vice-versa. SCIM formatted physical mailing addresses are represented as strings with embedded new lines, whereas LDAP uses the $ character to separate address lines. This transformation interprets new lines in SCIM values as address line separators.
- `com.unboundid.scim.ldap.TelephoneNumberTransformation`. Transforms LDAP Telephone Number syntax (E.123) to RFC3966 format and vice-versa.

You can also write your own transformations using the SCIM API described in the following section.

Mapping LDAP entries to SCIM using the SCIM-LDAP API

In addition to the SCIM SDK, PingDirectoryProxy Server provides a library called SCIM-LDAP, which provides facilities for writing custom transformations and more advanced mapping.

You can add the SCIM-LDAP library to your project using the following dependency:

```
<dependency>
    <groupId>com.unboundid.product.scim</groupId>
    <artifactId>scim-ldap</artifactId>
    <version>1.5.0</version>
</dependency>
```

Create your custom transformation by extending the `com.unboundid.scim.ldap.Transformation` class. Place your custom transformation class in a jar file in the server's `lib` directory.

ⓘ **Note:** The Identity Access API automatically maps LDAP attribute syntaxes to the appropriate SCIM attribute types. For example, an LDAP DirectoryString is automatically mapped to a SCIM string.

SCIM authentication

SCIM requests to the LDAP endpoints will support HTTP Basic Authentication and OAuth2 Authentication using a bearer token. There is existing support for this feature in the Directory Server and the Directory Proxy Server using the OAuthTokenHandler API (i.e., via a Server SDK extension, which requires some technical work to implement).

Note that our implementation only supports the HTTP Authorization header for this purpose; we do not support the form-encoded body parameter or URI query parameter mechanisms for specifying the credentials or bearer token.

SCIM logging

The Directory Proxy Server already provides a detailed HTTP log publisher to capture the SCIM and HTTP request details. To be able to correlate this data to the internal LDAP operations that are invoked behind the scenes, the Access Log Publisher will use "`origin=scim`" in access log messages that are generated by the SCIM servlet.

For example, you will see a message for operations invoked by replication:

```
[30/Oct/2012:18:45:10.490 -0500] MODIFY REQUEST conn=-3 op=190 msgID=191
origin="replication" dn="uid=user.3,ou=people,dc=example,dc=com"
```

Likewise for SCIM messages, you will see a message like this:

```
[30/Oct/2012:18:45:10.490 -0500] MODIFY REQUEST conn=-3 op=190 msgID=191
origin="scim" dn="uid=user.3,ou=people,dc=example,dc=com"
```

SCIM monitoring

There are two facilities that can be used to monitor the SCIM activity in the server.

- **HTTPConnectionHandlerStatisticsMonitorProvider** -- Provides statistics straight about total and average active connections, requests per connection, connection duration, processing time, invocation count, etc.
- **SCIMServletMonitorProvider** -- Provides high level statistics about request methods (POST, PUT, GET, etc.), content types (JSON, XML), and response codes, for example, "`user-patch-404:26`".

The LDAP object class endpoints are treated as their own resource types, so that for requests using the Identity Access API, there will be statistics, such as `person-get-200` and `inetorgperson-post-401`.

Configuring the Identity Access API

Once you have run the `<server-root>/config/scim-config-ds.dsconfig` script, the resources defined in the `scim-resources.xml` will be available as well as the Identity Access API. However, to allow SCIM access to the raw LDAP data, you must set a combination of configuration properties on the SCIM Servlet Extension using the **`dsconfig`** tool.

- **include-ldap-objectclass**. Specifies a multi-valued property that lists the object classes for entries that will be exposed. The object class used here will be the one that clients need to use when referencing Identity Access API resources. This property allows the special value "*" to allow all object classes. If "*" is used, then the SCIM servlet uses the same case used in the Directory Proxy Server LDAP Schema.
- **exclude-ldap-objectclass**. Specifies a multi-valued property that lists the object classes for entries that will not be exposed. When this property is specified, all object classes will be exposed except those in this list.
- **include-ldap-base-dn**. Specifies a multi-valued property that lists the base DNs that will be exposed. If specified, only entries under these base DNs will be accessible. No parent-child relationships in the DNs are allowed here.
- **exclude-ldap-base-dn**. Specifies a multi-valued property that lists the base DNs that will not be exposed. If specified, entries under these base DNs will not be accessible. No parent-child relationships in the DNs are allowed here.

Using a combination of these properties, SCIM endpoints will be available for all included object classes, just as if they were SCIM Resources defined in the `scim-resources.xml` file.

Configuring the Identity Access API

Steps

1. Ensure that you have run the `scim-config-ds.dsconfig` script to configure the SCIM interface. Be sure to enable the entryDN virtual attribute. See the Configure SCIM section for more information.
2. Set a combination of properties to allow the SCIM clients access to the raw LDAP data: `include-ldap-objectclass`, `exclude-ldap-objectclass`, `include-ldap-base-dn`, or `exclude-ldap-base-dn`.

```
$ bin/dsconfig set-http-servlet-extension-prop \
  --extension-name SCIM --set 'include-ldap-objectclass:*' \
```

```
  --set include-ldap-base-dn:ou=People,dc=example,dc=com
```

The SCIM clients now have access to the raw LDAP data via LDAP object class-based resources as well as core SCIM resources as defined in the `scim.resource.xml` file.

Disabling core SCIM resources

Steps

1. Open the `config/scim-resources.xml` file, and comment out or remove the `<resource>` elements that you would like to disable.
2. Disable and re-enable the HTTP Connection Handler, or restart the server to make the changes take effect. In general, changing the `scim-resources.xml` file requires a HTTP Connection Handler restart or server restart.

> ⓘ **Note:** When making other changes to the SCIM configuration by modifying the SCIM HTTP Servlet Extension using **`dsconfig`**, the changes take effect immediately without any restart required.

Verifying the Identity Access API configuration

Steps

- Perform a curl request to verify the Identity Access API configuration.

```
$ curl -k -u "cn=directory manager:password" \
  -H "Accept: application/json" \
  "https://example.com/top/56c9fd6b-f870-35ef-9959-691c783b7318?
     attributes=entryDN,uid,givenName,sn,entryUUID"
     {"schemas":
["urn:scim:schemas:core:1.0","urn:unboundid:schemas:scim:ldap:1.0"],
      "id":"56c9fd6b-f870-35ef-9959-691c783b7318",
      "meta":{"lastModified":"2013-01-11T23:38:26.489Z",
      "location":"https://example.com:443/v1/top/56c9fd6b-
f870-35ef-9959-691c783b7318"},
      "urn:unboundid:schemas:scim:ldap:1.0":{"givenName":["Rufus"],"uid":
["user.1"],
      "sn":["Firefly"],"entryUUID":["56c9fd6b-f870-35ef-9959-691c783b7318"],
      "entrydn":"uid=user.1,ou=people,dc=example,dc=com"}}
```

Monitoring the SCIM servlet extension

The SCIM SDK provides a command-line tool, `scim-query-rate`, that measures the SCIM query performance for your extension. The SCIM extension also exposes monitoring information for each SCIM resource, such as the number of successful operations per request, the number of failed operations per request, the number of operations with XML or JSON to and from the client. Finally, the Directory Proxy Server automatically logs SCIM-initiated LDAP operations to the default File-based Access Logger. These operations will have an `origin='scim'` attribute to distinguish them from operations initiated by LDAP clients. You can also create custom logger or request criteria objects that can track incoming HTTP requests, which the SCIM extension rewrites as internal LDAP operations.

Testing SCIM query performance

You can use the `scim-query-rate` tool, provided in the SCIM SDK, to test query performance, by performing repeated resource queries against the SCIM server.

The `scim-query-rate` tool performs searches using a query filter or can request resources by ID. For example, you can test performance by using a filter to query randomly across a set of one million users with eight concurrent threads. The user resources returned to the client in this example is in XML format and includes the `userName` and `name` attributes.

```
scim-query-rate --hostname server.example.com --port 80 \
--authID admin --authPassword password --xml \
--filter 'userName eq "user.[1-1000000]"' --attribute userName \
--attribute name --numThreads 8
```

You can request resources by specifying a resource ID pattern using the `--resourceID` argument as follows:

```
scim-query-rate --hostname server.example.com --port 443 \
--authID admin --authPassword password --useSSL --trustAll\
--resourceName User \
--resourceID 'uid=user.[1-150000],ou=people,dc=example,dc=com'
```

The `scim-query-rate` tool reports the error `"java.net.SocketException: Too many open files"` if the open file limit is too low. You can increase the open file limit to increase the number of file descriptors.

About the HTTP log publishers

HTTP operations may be logged using either a Common Log File HTTP Operation Log Publisher or a Detailed HTTP Operation Log Publisher. The Common Log File HTTP Operation Log Publisher is a built-in log publisher that records HTTP operation information to a file using the W3C common log format. Because the W3C common log format is used, logs produced by this log publisher can be parsed by many existing web analysis tools.

Log messages are formatted as follows:

- IP address of the client.
- RFC 1413 identification protocol. The Ident Protocol is used to format information about the client.
- The user ID provided by the client in an Authorization header, which is typically available server-side in the REMOTE_USER environment variable. A dash appears in this field if this information is not available.
- A timestamp, formatted as '"['dd/MM/yyyy:HH:mm:ss Z']"'
- Request information, with the HTTP method followed by the request path and HTTP protocol version.
- The HTTP status code value.
- The content size of the response body in bytes. This number does not include the size of the response headers.

The HTTP Detailed Access Log Publisher provides more information than the common log format in a format that is familiar to administrators who use the File-Based Access Log Publisher.

The HTTP Detailed Access Log Publisher generates log messages such as the following. The lines have been wrapped for readability.

```
[15/Feb/2012:21:17:04 -0600] RESULT requestID=10834128
from="10.2.1.114:57555" method="PUT"
url="https://10.2.1.129:443/Aleph/Users/6272c691-
38c6-012f-d227-0dfae261c79e" authorizationType="Basic"
requestContentType="application/json" statusCode=200
etime=3.544 responseContentLength=1063
redirectURI="https://server1.example.com:443/Aleph/Users/6272c691-38c6-012f-
d227-0dfae261c79e"
responseContentType="application/json"
```

In this example, only default log publisher properties are used. Though this message is for a RESULT, it contains information about the request, such as the client address, the request method, the request URL, the authentication method used, and the Content-Type requested. For the response, it includes the response length, the redirect URI, the Content-Type, and the HTTP status code.

You can modify the information logged, including adding request parameters, cookies, and specific request and response headers. For more information, refer to the **`dsconfig`** command-line tool help.

Monitoring resources using the SCIM extension

The monitor provider exposes the following information for each resource:

- Number of successful operations per request type (such as GET, PUT, and POST).
- Number of failed operations and their error codes per request type.
- Number of operations with XML or JSON from client.
- Number of operations that sent XML or JSON to client.

In addition to the information about the user-defined resources, monitoring information is also generated for the schema, service provider configuration, and monitor resources. The attributes of the monitor entry are formatted as follows:

```
{resource name}-resource-{request type}-{successful or error status code}
```

You can search for one of these monitor providers using an **`ldapsearch`** such as the following:

```
$ bin/ldapsearch --port 1389 bindDN uid=admin,dc=example,dc=com \
  --bindPassword password --baseDN cn=monitor \
  --searchScope sub "(objectclass=scim-servlet-monitor-entry)"
```

For example, the following monitor output was produced by a test environment with three distinct SCIM servlet instances, Aleph, Beth, and Gimel. Note that the first instance has a custom resource type called host.

```
$ bin/ldapsearch --baseDN cn=monitor \
  '(objectClass=scim-servlet-monitor-entry)'
dn: cn=SCIM Servlet (SCIM HTTP Connection Handler),cn=monitor
objectClass: top
objectClass: ds-monitor-entry
objectClass: scim-servlet-monitor-entry
objectClass: extensibleObject
cn: SCIM Servlet (SCIM HTTPS Connection Handler) [from
  ThirdPartyHTTPServletExtension:SCIM (Aleph)]
ds-extension-monitor-name: SCIM Servlet (SCIM HTTPS Connection Handler)
ds-extension-type: ThirdPartyHTTPServletExtension
ds-extension-name: SCIM (Aleph)
version: 1.2.0
build: 20120105174457Z
revision: 820
schema-resource-query-successful: 8
schema-resource-query-401: 8
schema-resource-query-response-json: 16
user-resource-delete-successful: 1
user-resource-put-content-xml: 27
user-resource-query-response-json: 3229836
user-resource-put-403: 5
user-resource-put-content-json: 2
user-resource-get-401: 1
user-resource-put-response-json: 23
user-resource-get-response-json: 5
user-resource-get-response-xml: 7
user-resource-put-400: 2
user-resource-query-401: 1141028
user-resource-post-content-json: 1
user-resource-put-successful: 22
user-resource-post-successful: 1
user-resource-delete-404: 1
user-resource-query-successful: 2088808
user-resource-get-successful: 10
user-resource-put-response-xml: 6
user-resource-get-404: 1
user-resource-delete-401: 1
```

```
user-resource-post-response-json: 1
host-resource-query-successful: 5773268
host-resource-query-response-json: 11576313
host-resource-query-400: 3
host-resource-query-response-xml: 5
host-resource-query-401: 5788152
dn: cn=SCIM Servlet (SCIM HTTP Connection Handler),cn=monitor
objectClass: top
objectClass: ds-monitor-entry
objectClass: scim-servlet-monitor-entry
objectClass: extensibleObject
cn: SCIM Servlet (SCIM HTTPS Connection Handler) [from
  ThirdPartyHTTPServletExtension:SCIM (Beth)]
ds-extension-monitor-name: SCIM Servlet (SCIM HTTPS Connection
  Handler)
ds-extension-type: ThirdPartyHTTPServletExtension
ds-extension-name: SCIM (Beth)
version: 1.2.0
build: 20120105174457Z
revision: 820
serviceproviderconfig-resource-get-successful: 3
serviceproviderconfig-resource-get-response-json: 2
serviceproviderconfig-resource-get-response-xml: 1
schema-resource-query-successful: 8
schema-resource-query-401: 8
schema-resource-query-response-json: 16
group-resource-query-successful: 245214
group-resource-query-response-json: 517841
group-resource-query-400: 13711
group-resource-query-401: 258916
user-resource-query-response-json: 107876
user-resource-query-400: 8288
user-resource-get-400: 33
user-resource-get-response-json: 1041
user-resource-get-successful: 2011
user-resource-query-successful: 45650
user-resource-get-response-xml: 1003
user-resource-query-401: 53938
dn: cn=SCIM Servlet (SCIM HTTP Connection Handler),cn=monitor
objectClass: top
objectClass: ds-monitor-entry
objectClass: scim-servlet-monitor-entry
objectClass: extensibleObject
cn: SCIM Servlet (SCIM HTTPS Connection Handler) [from
  ThirdPartyHTTPServletExtension:SCIM (Gimel)]
ds-extension-monitor-name: SCIM Servlet (SCIM HTTPS Connection
  Handler)
ds-extension-type: ThirdPartyHTTPServletExtension
ds-extension-name: SCIM (Gimel)
version: 1.2.0
build: 20120105174457Z
revision: 820
schema-resource-query-successful: 1
schema-resource-query-401: 1
schema-resource-query-response-json: 2
user-resource-query-successful: 65
user-resource-get-successful: 4
user-resource-get-response-json: 6
user-resource-query-response-json: 132
user-resource-get-404: 2
user-resource-query-401: 67
```

Managing the SCIM 2.0 Servlet Extension

The PingDirectoryProxy Server provides a servlet extension that implements version 2.0 of the System for Cross-domain Identity Management (SCIM) specification. The SCIM 2.0 extension does not replace the existing SCIM 1.1 extension, and there are significant differences in how they are configured..

ⓘ **Note:** SCIM configuration (attribute mappings, etc.) should be done on Proxy instead of on the backend PD servers.

Supported SCIM 2.0 Endpoints

PingDirectoryProxy Server supports the following SCIM 2.0 endpoints:

Endpoint	Description	Supported HTTP methods
/ServiceProviderConfig	Provides metadata that indicates the PingDirectoryProxy Server's authentication scheme (which is always OAuth 2.0) and its support for features that are optional in the SCIM standard. This endpoint is a metadata endpoint and is not subject to ACI restrictions.	`GET`
/Schemas	Lists the SCIM schemas that are configured for use on PingDirectoryProxy Server, and defines the various attributes available to resource types. This endpoint is a metadata endpoint and is not subject to ACI restrictions.	`GET`
/Schemas/<schema>	Retrieves a specific SCIM schema, as specified by the schema ID. This endpoint is a metadata endpoint and is not subject to ACI restrictions.	`GET`
/ResourceTypes	Lists all of the SCIM resource types that are configured for use on PingDirectoryProxy Server. Clients can use this information to determine the endpoint, core schema, and extension schemas of any resource types that the server supports. This endpoint is a metadata endpoint and is not subject to ACI restrictions.	`GET`

Endpoint	Description	Supported HTTP methods
/ResourceTypes/<resourceType>	Retrieves a specific SCIM resource type, as specified by its ID. This endpoint is a metadata endpoint and is not subject to ACI restrictions.	`GET`
/<resourceType>	Creates a new resource (`POST`), or lists and filters resources (`GET`).	`GET`, `POST`
/<resourceType>/.search	Lists and filters resources. This is used if passing query parameters in the URL is undesirable.	`POST`
/<resourceType>/<resourceId>	Retrieves a single resource (`GET`), modifies a single resource (`PUT`, `PATCH`), or deletes a single resource (`DELETE`).	`GET`, `PUT`, `PATCH`, `DELETE`

Configuring SCIM 2.0 on Your Server
Creating Your Own SCIM 2 Application

The System for Cross-domain Identity Management (SCIM) is an open initiative designed to make moving identity data to, from, and between clouds standard, secure, fast, and easy. The SCIM SDK is a pre-packaged collection of libraries and extensible classes that provides developers with a simple, concrete API to interact with a SCIM service provider.

The SCIM 2 SDK is available for download at https://github.com/pingidentity/scim.

Authentication Requirements for SCIM 2.0 Requests

All SCIM requests on a PingDirectoryProxy Server must use OAuth 2.0 bearer token authentication. Bearer tokens are evaluated using the SCIM 2 servlet extension's configured Access Token Validators. Basic authentication is not supported.

In addition to requiring bearer tokens, the server will not process any SCIM requests unless it has at least one encryption-settings definition in its encryption-settings database. You can create the necessary definition(s) with the encryption-settings command-line tool. See *Encrypting Sensitive Data* on page 544 for more information.

ⓘ **Note:** Encryption settings need to be defined on the Proxy and all backend PD servers, and the encryption settings should match.

Defining Permissions for SCIM 2.0 Requests

Whether or not a SCIM request is executed on a server depends primarily on both the configured Access Control Instructions (ACIs) in the server and the scopes which are present in the provided `OAuth` bearer token used to authenticate the request.

ⓘ **Note:** ACIs need to be defined on the backend PD servers instead of on the Proxy server.

Internally, all SCIM 2.0 requests are processed using the cn=SCIM2 Servlet,cn=Root DNs,cn=config service account. Whether a requested operation is allowed depends on the ACIs that apply to the operation. The oauthscope bind rule is particularly useful for this, since it allows the administrator to use the supplied `OAuth` scopes in ACI logic.

Due to implementation details, access to the objectclass operational LDAP attribute is necessary for SCIM requests to properly execute. However, it is not advisable to give the service account access to objectclass on a global level. Instead, add the ACI granting objectclass access to the LDAP subtree you wish to expose to clients. See *Configuring permissions for SCIM 2.0 operations* on page 808 for an example of this.

ⓘ **Note:** ACIs that do not use the oauthscope bind rule can still apply to requested operations. For example, an ACI that grants unconditional read access to any authenticated LDAP user will also grant unconditional read access to SCIM requests regardless of the provided `OAuth` scopes, since the requests are processed through the service account.

SCIM 2.0 Components

Unlike the SCIM 1.1 servlet extension, the SCIM 2.0 system is configured through the Administrative Console or with the **`dsconfig`** command-line tool. The SCIM 2.0 system consists of the following components:

- SCIM resource types
- SCIM schemas
  - SCIM attributes
  - SCIM sub-attributes
- SCIM attribute mappings (mapping resource types only)
- Correlated LDAP Data Views

A SCIM resource type defines a class of resources, such as users or devices. Every SCIM resource type features at least one SCIM schema, which defines the attributes that are available to each resource. If enabled for use, a SCIM resource type must also have a designated LDAP structural objectclass as well as an associated base DN.

The two types of SCIM resource types, mapping and passthrough, differ based on the definitions of the SCIM schema the resource types use.

- An LDAP mapping SCIM resource type requires an explicitly defined SCIM schema with explicitly defined mappings of SCIM attributes to LDAP attributes. Use a mapping SCIM resource type to exercise detailed control over the SCIM schema and its attributes and mappings.
- An LDAP passthrough SCIM resource type, by contrast, does not use an explicitly defined SCIM schema. Instead, an implicit schema is generated dynamically, based on the underlying LDAP schema. Use a passthrough SCIM resource type when you need to get started quickly.

A SCIM schema defines a collection of SCIM attributes, grouped under an identifier called a schema URN. Each SCIM resource type possesses a single core schema and can feature schema extensions, which act as secondary attribute groupings that the schema URN namespaces. SCIM Schemas are defined independently of SCIM resource types, and multiple SCIM resource types can use a single SCIM schema as a core schema or schema extension.

A SCIM attribute defines an attribute that is available under a SCIM schema. The configuration for a SCIM attribute defines its data type, regardless of whether it is required, single-valued, or multi-valued. When a SCIM attribute consists of SCIM sub-attributes, it is defined as a complex attribute.

A SCIM attribute mapping defines the manner in which a SCIM resource type maps the attributes in its SCIM schemas to native LDAP attributes of the PingDirectoryProxy Server.

A Correlated LDAP Data View allows a single SCIM resource that consists of attributes that are retrieved from multiple LDAP entries (see *Correlated LDAP data views* on page 803).

Correlated LDAP data views

Correlated LDAP data views can be attached to a SCIM resource type to allow that resource type to use attributes from other LDAP entries. A SCIM resource type can have multiple data views configured.

Correlated LDAP data views share many configuration settings with SCIM resource types. The following exceptions are below:

- primary-correlation-attribute: The LDAP attribute from the SCIM Resource Type whose value will be used to match entries in the data view. This property must be defined.
- secondary-correlation-attribute: The LDAP attribute from the data view whose value will be matched with the primary-correlation-attribute. This property must be defined.
- ldap-correlation-attribute-pair: Optional correlation attribute pairs. If these are configured, entries between the SCIM resource type and the correlated LDAP data view must also have matching values for the specified attributes for them to be returned together.

ⓘ **Note:** A a correlated LDAP Data View cannot provide attributes from more than one LDAP entry at a time. If there are multiple LDAP entries with secondary-correlation-attribute values that match the primary-correlation-attribute from the SCIM resource type's entry, an error will be thrown.

If the correlated LDAP data view is attached to a Passthrough SCIM resource type, its attributes will appear as schema extensions, similar to the following:

```
{
      ...
 "uid": [
     "user.8"
 ],
 "entryDN": "uid=user.8,ou=people,dc=example,dc=com",
 "urn:pingidentity:schemas:correlated:Document": {
         "documentIdentifier": [
             "user.8"
         ],
         "description": [
             "This is the description for the document user.8 under
 ou=Documents,dc=example,dc=com."
         ],
         ...
 },
 ...
 "schemas": [
     "urn:pingidentity:schemas:correlated:Document",
     "urn:pingidentity:schemas:passthrough:PassthroughUsers"
 ]
}
```

If the correlated LDAP data view is attached to a Mapping SCIM resource type, its attributes must be mapped to a schema used by the SCIM resource type. This is done through the correlated-ldap-data-view property in a SCIM attribute mapping.

Configuring an LDAP mapped SCIM resource type

About this task

This example shows how to add a simple mapping SCIM 2.0 resource type backed by the inetOrgPerson LDAP objectclass to a PingDirectoryProxy Server deployment.

Steps

1. Set up the PingDirectory Server backend server. For this example, the default settings should be used, meaning that sample data will be used and that data encryption has been configured. After the server has been set up, export the encryption-settings definition with the related tool's export subcommand:

   ```
   encryption-settings export --output-file exported-key
   ```

2. Set up the PingDirectoryProxy Server, making sure to import the encryption-settings definition file that was created in the previous step. Then use the **`create-initial-proxy-config`** tool to configure the LDAP external server.
3. Create the SCIM schema that the resource type will use:

```
dsconfig create-scim-schema \
--schema-name urn:pingidentity:schemas:User:1.0 \
--set display-name:User
```

4. Under this schema, add the following SCIM attributes.

```
dsconfig create-scim-attribute \
--schema-name urn:pingidentity:schemas:User:1.0 \
--attribute-name displayName
dsconfig create-scim-attribute \
--schema-name urn:pingidentity:schemas:User:1.0 \
--attribute-name name \
--set type:complex
dsconfig create-scim-subattribute \
--schema-name urn:pingidentity:schemas:User:1.0 \
--attribute-name name \
--subattribute-name familyName
dsconfig create-scim-subattribute \
--schema-name urn:pingidentity:schemas:User:1.0 \
--attribute-name name \
--subattribute-name formatted
dsconfig create-scim-attribute \
--schema-name urn:pingidentity:schemas:User:1.0 \
--attribute-name userName
```

5. Create the LDAP mapping SCIM resource type on the PingDirectoryProxy Server.

```
dsconfig create-scim-resource-type \
--type-name Users \
--type ldap-mapping \
--set enabled:true \
--set endpoint:Users \
--set structural-ldap-objectclass:inetOrgPerson \
--set include-base-dn:ou=People,dc=example,dc=com \
--set lookthrough-limit:500 \
--set core-schema:urn:pingidentity:schemas:User:1.0
```

6. Run the followihg commands to create the SCIM attribute mappings.

```
dsconfig create-scim-attribute-mapping \
--type-name Users \
--mapping-name displayName \
--set scim-resource-type-attribute:displayName \
--set ldap-attribute:displayName
dsconfig create-scim-attribute-mapping \
--type-name Users \
--mapping-name name.formatted \
--set scim-resource-type-attribute:name.formatted \
--set ldap-attribute:cn \
--set searchable:true
dsconfig create-scim-attribute-mapping \
--type-name Users \
--mapping-name name.familyName \
--set scim-resource-type-attribute:name.familyName \
--set ldap-attribute:sn \
--set searchable:true
dsconfig create-scim-attribute-mapping \
```

```
--type-name Users \
--mapping-name userName \
--set scim-resource-type-attribute:userName \
--set ldap-attribute:uid \
--set searchable:true
```

7. Configure the SCIM2 HTTP Servlet Extension to use a Mock Access Token Validator. Note that Mock Access Token Validators should never be used in production environments or with sensitive data.

```
dsconfig create-access-token-validator \
--validator-name "SCIM2 Mock Validator" \
--type mock \
--set enabled:true
dsconfig set-http-servlet-extension-prop \
--extension-name SCIM2 \
--set "access-token-validator:SCIM2 Mock Validator"
```

8. Send the following request to the PingDirectoryProxy Server's SCIM /ResourceTypes endpoint to confirm that the new resource type has been added. The HTTP port may vary depending on how the deployment was configured.

```
curl -k -X GET \
https://localhost:8443/scim/v2/ResourceTypes \
-H 'Authorization: Bearer {"active":true}'
```

9. The following JSON object should appear in the response in the “Resources” array:

```
{
...
"Resources": [{
"schemas":["urn:ietf:params:scim:schemas:core:2.0:ResourceType"],
"id":"Users",
"name":"Users",
"endpoint":"Users",
"schema":"urn:pingidentity:schemas:Users:1.0",
"meta":{
"resourceType":"ResourceType",
"location":"https://localhost:8443/scim/v2/ResourceTypes/Users"
}
}]
...
}
```

Configuring Permissions for SCIM 2.0 Operations Proxy

About this task

This example shows how to correctly configure permissions so POST requests with the "userAdd" scope will succeed on a PingDirectoryProxy Server deployment. This example assumes that you have set up an LDAP mapping SCIM 2.0 Resource Type for the inetOrgPerson objectclass (see *Configuring an LDAP Mapping SCIM 2.0 resource type* on page 803).

Steps

1. If the SCIM Resource Type being targeted already has a value for the create-dn-pattern property, skip to step 2. Otherwise, run the following dsconfig command on the PingDirectoryProxy Server.

```
dsconfig set-scim-resource-type-prop \
--type-name Users \
--set create-dn-pattern:entryUUID=generated,ou=People,dc=example,dc=com
```

2. Send the following request to the PingDirectoryProxy Server's SCIM /Users endpoint. The HTTP port may vary depending on how the deployment was configured.

```
curl -k -X POST \
https://localhost:8443/scim/v2/Users/ \
-H 'Authorization: Bearer {"active":true}' \
-H 'Content-type: application/json' \
--data '{"username":"user.test", "name":{"formatted":"Test",
"familyName":"User"}, "schemas":["urn:pingidentity:schemas:User:1.0"]}'
```

   The response from the server should have a status of 403 and should contain a correlation ID, similar to the following:

```
{
"schemas":["urn:ietf:params:scim:api:messages:2.0:Error"],
"status":"403",
"detail":"Request failed:
correlationID='faa707b3-5d48-42e6-9e78-2c8dbb1e2cac'"
}
```

   This is the expected response since this SCIM request does not have the permission needed to write to an entry. See *Troubleshoot the SCIM 2.0 Servlet Extension* on page 1206 for instructions on viewing the full server error message.

3. An ACI should now be added to the backend server's ou=People,dc=example,dc=com subtree. This ACI grants permission to add entries to the said subtree as long as the SCIM request contains the userAdd scope. The following **`ldapmodify`** command for creating the ACI must be run on the backend PingDirectory Server server and not the PingDirectoryProxy Server endpoint . Note that this ACI does not grant write access to attributes, which means modify operations will not succeed. See *Overview of access control* on page 567 for more information on configuring ACIs,.

```
$ ldapmodify
dn:ou=People,dc=example,dc=com
changetype:modify
add:aci
aci:(version 3.0; acl "ACI for userAdd scope"; allow (add)
oauthscope="userAdd";)
```

4. Send the POST request to the SCIM / Users endpoint again, this time including the userAdd scope in the bearer token:

```
curl -k -X POST \
https://localhost:8443/scim/v2/Users \
-H 'Authorization: Bearer {"active":true, "scope":"userAdd"}' \
-H 'Content-type: application/json' \
--data '{"username":"user.test", "name":{"formatted":"Test",
"familyName":"User"}, "schemas":["urn:pingidentity:schemas:User:1.0"]}'
```

   The response from the server should now contain the created SCIM resource, which should also contain values for the name and username attributes, similar to the following:

```
{
"name":{
"familyName":"User",
"formatted":"Test"
},
"username":"user.test",
"id":"6f9a89b8-e766-478c-9667-def049daf6bc",
"meta":{
"resourceType":"Users",
"location":"https://localhost:8443/scim/v2/Users/6f9a89b8-e766-478c-9667-
```

```
def049daf6bc"
},
"schemas":["urn:pingidentity:schemas:User:1.0"]
}
```

SCIM 2.0 Searches

To prevent exhausting server resources, we recommend capping the total number of resources that are matched by a search. The configuration for each SCIM 2.0 resource type contains a lookthrough-limit property that defines this limit (the default lookthrough-limit value is 500).

If a search request exceeds the lookthrough limit, the client receives a 400 response with an error message similar to the following:

```
{
 "detail": "The search request matched too many results",
 "schemas": ["urn:ietf:params:scim:api:messages:2.0:Error"],
 "scimType": "tooMany",
 "status": "400"
}
```

To prevent this error, you have these options:

- The client must refine its search filter to return fewer matches.
- Configure paged searches as explained in *Using paged SCIM searches* on page 1202.

Using paged SCIM searches

When searching large data sets, the results can be numerous and produce errors about a request matching too many results relative to the lookthrough limit. Paged searches avoid these errors and also reduce memory utilization.

Before you begin

The paged SCIM searches feature is not available for entry-balanced data sets.

To use paged SCIM searches, your SCIM service's backend servers must be LDAP directory servers.

Complete the following one-time operation. You only need to run the command one time per PingDirectoryProxy Server. If you are not sure whether you have run the command, you can run it again safely.

```
$ dsconfig set-request-processor-prop \
--processor-name <proxying-request-processor> \
--set supported-control-oid:2.16.840.1.113730.3.4.9 \
--set supported-control-oid:1.2.840.113556.1.4.473
```

where *<proxying-request-processor>* is the request processor handling the entries targeted by the search.

About this task

PingDirectoryProxy Server does SCIM searches using LDAP requests. After you complete the steps below, PingDirectoryProxy Server creates LDAP requests that include request controls that ask the backend servers to sort and page the search results before returning the results. These request controls are marked noncritical, meaning that if the backend server cannot page the results, the backend server still returns the results. In this case, PingDirectoryProxy Server handles the sorting and paging itself.

If your SCIM searches result in an error because the request matched too many results, as discussed in *SCIM 2.0 Searches* on page 1202, you can avoid the error by using paged searches.

Complete the following steps for each search.

Steps

1. Decide your SCIM search.

   ⓘ **Note:** To get paged results, your search must include at least one of these parameters: startIndex, count, or sortBy.

   For example, your search might look like the following search.

   ```
   https://<proxy-hostname>:<proxy-port>/scim/v2/Users/?filter=st eq
    "TX"&sortBy=sn&sortOrder=ascending
   ```

   Here is the corresponding encoded version.

   ```
   https://<proxy-hostname>:<proxy-port>/scim/v2/Users/?filter=st%20eq
   %20%22TX%22&sortBy=sn&sortOrder=ascending
   ```

   On your PingDirectoryProxy Server, collect some information to use later. Find the SCIM resource type, `structural-ldap-objectclass`, `include-base-dn`, and `include-filter` values by running this command.

   ```
   $ dsconfig get-scim-resource-type-prop --type-name <SCIM-resource-type-
   name> \
   --property structural-ldap-objectclass \
   --property include-base-dn \
   --property include-filter
   ```

2. On each backend server, complete the following steps.
   a. Create a Virtual List View (VLV) index for your search.

      Each SCIM search that you want to produce paged results must have its own VLV index.

      Create this index using `dsconfig create-local-db-vlv-index` with the following options.

Option	Description
`--index-name`	Names the index.
`--backend-name`	Specifies the name of the local database backend in which to place the index. The default database backend for PingDirectory is userRoot.
`--set base-dn`	Specifies the desired base dn. This value must match the value of the `include-base-dn` property that you found in the previous step.
`--set scope`	Is always `whole-subtree`.

Option	Description
`--set filter`	Specifies the filter. Specify "`(objectclass=`*`<name-of-SCIM-resource-type-objectclass>`*`)`" where *`<name-of-SCIM-resource-type-objectclass>`* is the name of the objectclass used by the SCIM resource type property, which you found in the previous step. If the SCIM resource type has the `include-filter` property set, also specify that property value in the filter. For example, if the filter for the objectclass is `(objectclass=inetorgperson)` and the `include-filter` value is `(st=CA)`, specify the `--set filter` argument as "`(&(objectclass=inetorgperson)(st=CA))`". Specify the LDAP attributes for all the components of your SCIM search filter. For example, if a mapping SCIM resource type maps the LDAP attribute `st` to the SCIM attribute `address.region` and the SCIM search filter requires that `address.region eq TX`, then this filter must include `(st = TX)` instead of `(address.region = TX)`.
`--set sort-order`	Specifies whether to sort ascending (+) or descending (-) and the LDAP attribute to sort by. If the SCIM search does not specify the `sortBy` parameter, specify the sort order as `+entryUUID`.

Recall the original, decoded SCIM search, shown here.

```
https://<proxy-hostname>:<proxy-port>/scim/v2/Users/?filter=st eq
 "TX"&sortBy=sn&sortOrder=ascending
```

For example, to create a VLV index for that search, run the following command.

```
$ dsconfig create-local-db-vlv-index --index-name sn \
--backend-name userRoot --set base-dn:ou=people,dc=example,dc=com \
--set scope:whole-subtree \
--set filter:"(&(objectclass=inetorgperson)(st=TX))" --set sort-order:
+sn
```

b. Stop the server. Rebuild the index. Start the server. Run the `rebuild-index` command specifying the baseDN and the name of the index.

```
$ rebuild-index --baseDN <baseDN-value> --index <name-of-index>
```

For example, run these commands.

```
$ stop-server
$ rebuild-index --baseDN dc=example,dc=com --index vlv.sn
$ start-server
```

3. Run your SCIM search filter.

> ⓘ **Note:**
>
> The search can include only the filter you specified with `--set filter` in the earlier step without the "`(objectclass=`*`<name-of-SCIM-resource-type-objectclass>`*`)`" portion.

In addition to the Virtual List View request control, PingDirectoryProxy Server adds a Server Side request control to the LDAP request. These request controls require certain parameters be set. To satisfy this requirement, PingDirectoryProxy Server uses the following parameters. If the client does not provide values for one of the parameters, the search uses the corresponding default value shown in the following table.

Parameter	Default
startIndex	1
count	The value of the `lookthrough-limit` property of the SCIM resource type being searched. That default is 500.
sortBy	entryUUID With this default, the results appear unsorted.
sortOrder	ascending

SCIM 2.0 PATCH Operations

When using a `PATCH` request to modify a SCIM 2.0 resource that has one or more required SCIM 2.0 attributes, the requester must have permissions to read the values of these required attributes in addition to write permissions for the attributes being modified, even if the `PATCH` request does not alter said requirements.

For example, assume we want to modify an LDAP Mapping SCIM 2.0 resource type using the following schema definition, where uid and cn are mapped to their LDAP equivalents:

```
{
 "schemas": ["urn:ietf:params:scim:schemas:core:2.0:Schema"],
 "id": "urn:test:schema:person",
 "attributes": [
 {
 "name": "uid",
 "type": "string",
 "multiValued": false,
 "required": true,
 "caseExact": false,
 "mutability": "readWrite",
 "returned": "default",
 "uniqueness": "none"
 },
 {
 "name": "cn",
 "type": "string",
 "multiValued": false,
 "required": false,
 "caseExact": false,
 "mutability": "readWrite",
 "returned": "default",
 "uniqueness": "none"
 }
  ],
```

```
  ...
   }
```

The following `PATCH` operation will fail if the SCIM 2 service account does not have access to both uid and cn:

```
{
 "schemas": ["urn:ietf:params:scim:api:messages:2.0:PatchOp"],
 "Operations":[{
 "op":"add",
 "path":"cn",
 "value": "new cn value”
 }]
  }
```

Troubleshoot the SCIM 2.0 Servlet Extension

For security reasons, error messages specifically regarding LDAP systems are suppressed and do not appear in the HTTP responses from the server. Instead, you will see something similar to the following:

```
{
 "schemas": [
 "urn:ietf:params:scim:api:messages:2.0:Error"
 ],
 "status": "400",
 "detail": "Request failed:
 correlationID='073eb1a8-8c51-48b3-83a0-380e1d4b4ab9'"
}
```

To view these messages, the Debug Trace Logger needs to be enabled. You can do this through the Administrative Console or with the following **`dsconfig`** command:

```
dsconfig set-log-publisher-prop --publisher-name "Debug Trace Logger" \
   --set enabled:true --add scim-message-type:error
```

After the Debug Trace Logger is enabled, the server will begin logging information related to SCIM operations to the file /logs/debug-trace , which will look somewhat like the following:

```
[09/Jun/2020:05:23:10.992 -0500] HTTP REQUEST requestID=3
correlationID="073eb1a8-8c51-48b3-83a0-380e1d4b4ab9" product="Ping Identity
Directory Server" instanceName="example" startupID="Xt9fJg==" threadID=173
from=[0:0:0:0:0:0:0:1]:53978 method=POST
url="https://0:0:0:0:0:0:0:1:9443/scim/v2/Users"
```

Note the presence of correlationID in these messages. By matching the id in the HTTP responses to the messages in the debug-trace log, the appropriate LDAP error message can be determined.

Disable the SCIM 2.0 Servlet Extension

If you do not need to expose data through the SCIM 2 servlet, you can disable the SCIM 2.0 servlet extension by removing the SCIM2 HTTP servlet extension from the HTTPS connection handler, or from any other HTTP connection handler, and restart the handler:

```
dsconfig set-connection-handler-prop \
 --handler-name "HTTPS Connection Handler" \
 --remove http-servlet-extension:SCIM2 \
 --set enabled:false

 dsconfig set-connection-handler-prop \
```

```
  --handler-name "HTTPS Connection Handler" \
  --set enabled:true
```

Managing Server SDK Extensions

The PingDirectoryProxy Server provides support for any custom extensions that you create using the Server SDK. This chapter summarizes the various features and extensions that can be developed using the Server SDK.

About the Server SDK

You can create extensions that use the Server SDK to extend the functionality of your Directory Proxy Server. Extension bundles are installed from a .zip archive or a file system directory. You can use the **`manage-extension`** tool to install or update any extension that is packaged using the extension bundle format. It opens and loads the extension bundle, confirms the correct extension to install, stops the server if necessary, copies the bundle to the server install root, and then restarts the server.

ⓘ **Note:** The **`manage-extension`** tool may only be used with Java extensions packaged using the extension bundle format. Groovy extensions do not use the extension bundle format. For more information, see the "Building and Deploying Java-Based Extensions" section of the Server SDK documentation, which describes the extension bundle format and how to build an extension.

Available types of extensions

The Server SDK provides support for creating a number of different types of extensions for Ping Identity Server Products, including the PingDirectory Server, PingDirectoryProxy Server, and PingDataSync Server. Some of those extensions include:

Cross-Product Extensions

- Access Loggers
- Alert Handlers
- Error Loggers
- Key Manager Providers
- Monitor Providers
- Trust Manager Providers
- OAuth Token Handlers
- Manage Extension Plugins

PingDirectory Server Extensions

- Certificate Mappers
- Change Subscription Handlers
- Extended Operation Handlers
- Identity Mappers
- Password Generators
- Password Storage Schemes
- Password Validators
- Plugins
- Tasks
- Virtual Attribute Providers

PingDirectoryProxy Server Extensions

- LDAP Health Checks

- Placement Algorithms
- Proxy Transformations

PingDataSync Server Extensions

- JDBC Sync Sources
- JDBC Sync Destinations
- LDAP Sync Source Plugins
- LDAP Sync Destination Plugins
- Sync SourcesSync Destinations
- Sync Pipe Plugins

For more information on the Server SDK, see the documentation available in the SDK build.

Command-Line Tools

The PingDirectoryProxy Server provides a full suite of command-line tools necessary to administer the server. The command-line tools are available in the `bin` directory for UNIX or Linux systems and `bat` directory for Microsoft Windows systems.

Available command-line tools

Directory Proxy Server provides the following command-line tools, which can be run directly in interactive, noninteractive, or script mode.

Tools Help

For	Use this option	Example
Information about arguments and subcommands Usage examples	`--help`	**`dsconfig --help`**
A list of subcommands	`--help-subcommands`	**`dsconfig --help-subcommands`**
More information about a subcommand	`--help` with the subcommand	**`dsconfig list-log-publishers --help`**

ⓘ **Note:** For detailed information and examples of the command-line tools, see the *Ping Identity Directory Proxy Server Command-Line Tool Reference*.

Command-Line Tools

authrate	Perform repeated authentications against an LDAP directory server, where each authentication consists of a search to find a user followed by a bind to verify the credentials for that user.
backup	Run full or incremental backups on one or more Directory Proxy Server backends. This tool also supports the use of a properties file to pass predefined command-line arguments. See *Saving Options in a File* for more information.
base64	Encode raw data using the base64 algorithm or decode base64-encoded data back to its raw representation.

collect-support-data	Collect and package system information useful in troubleshooting problems. The information is packaged as a zip archive that can be sent to a technical support representative.
config-diff	Compare Directory Proxy Server configurations and produce a **`dsconfig`** batch file needed to bring the source inline with the target.
create-initial-proxy-config	Create an initial Directory Proxy Server configuration.
create-rc-script	Create an RC script to start, stop, and restart the Directory Proxy Server on UNIX-based systems.
create-systemd-script	Create a **`systemd`** script to start and stop the Directory Proxy Server on Linux-based systems.
deliver-one-time-password	Generate and deliver a one-time password to a user through some out-of-band mechanism. That password can then be used to authenticate using the UNBOUNDID-DELIVERED-OTP SASL mechanism.
deliver-password-reset-token	Generate and deliver a single-use token to a user through some out-of-band mechanism. The user can provide that token to the password modify extended request in lieu of the user's current password in order to select a new password.
dsconfig	View and edit the Directory Proxy Server configuration.
dsjavaproperties	Configure the JVM arguments used to run the Directory Proxy Server and associated tools. Before launching the command, edit the properties file located in `config/java.properties` to specify the desired JVM options and JAVA_HOME environment variable.
dump-dns	Obtain a listing of all of the DNs for all entries below a specified base DN in the Directory Proxy Server.
encrypt-file	Encrypt or decrypt data using a key generated from a user-supplied passphrase, a key generated from an encryption settings definition, or a key shared among servers in the topology. The data to be processed can be read from a file or standard input, and the resulting data can be written to a file or standard output. You can use this command to encrypt and subsequently decrypt arbitrary data, or to decrypt encrypted backups, LDIF exports, and log files generated by the server.
encryption-settings	Manage the server encryption settings database.
enter-lockdown-mode	Request that the Directory Proxy Server enter lockdown mode, during which it only processes operations requested by users holding the `lockdown-mode` privilege.
export-reversible-passwords	Requests that the server export entries from a specified backend in LDIF form, including clear-text representations of any passwords encoded with a reversible storage scheme. This tool can only be used over a secure connection and when authenticated as a user with the permit-export-reversible-passwords privilege. The output is encrypted using a key generated from either a user-supplied passphrase or an encryption settings definition.
generate-totp-shared-secret	Generate a shared secret that may be used to generate time-based one-time password (TOTP) authentication codes for use in authenticating with the UNBOUNDID-TOTP SASL mechanism, or in conjunction with the validate TOTP password extended operation.

global-index-size	Estimate the size in memory of one or more global indexes from the actual number of keys, the configured number of keys and the average key size. The estimate could be slightly higher or lower than the actual size. An estimate can be provided for more than one index in one invocation by providing multiple sets of options.
identify-references-to-missing-entries	Identify entries containing one or more attributes that reference entries that do not exist. This might require the ability to perform unindexed searches and/or the ability to use the simple paged results control.
identify-unique-attribute-conflicts	Identify unique attribute conflicts. In particular, it identifies values of one or more attributes that are supposed to exist only in a single entry but are found in multiple entries.
indent-ldap-filter	Parse a provided LDAP filter string and displays it as a multi-line form that makes it easier to understand its hierarchy and embedded components. If possible, it also simplifies the provided filter in certain ways (for example, by removing unnecessary levels of hierarchy, like an AND embedded in an AND).
ldap-debugger	Intercept and decode LDAP communication.
ldap-diff	Compare the contents of two LDAP directory server servers.
ldap-result-code	Display and query LDAP result codes.
ldapcompare	Perform compare operations in an LDAP directory server. Compare operations might be used to efficiently determine whether a specified entry has a given value.
ldapdelete	Delete one or more entries from an LDAP directory server. You can provide the DNs of the entries to delete using named arguments, as trailing arguments, from a file, or from standard input. Alternatively, you can identify entries to delete using a search base DN and filter.
ldapmodify	Apply a set of add, delete, modify, and/or modify DN operations to a directory server. Supply the changes to apply in LDIF format, either from standard input or from a file specified with the `ldifFile` argument. Change records must be separated by at least one blank line.
ldappasswordmodify	Update the password for a user in an LDAP directory server using the password modify extended operation (as defined in RFC 3062), a standard LDAP modify operation, or an Active Directory-specific modification.
ldapsearch	Process one or more searches in an LDAP directory server.
ldif-diff	Compare the contents of two files containing LDIF entries. The output will be an LDIF file containing the add, delete, and modify change records needed to convert the data in the source LDIF file into the data in the target LDIF file.
ldifmodify	Apply a set of changes (including add, delete, modify, and modify DN operations) to a set of entries contained in an LDIF file. The changes will be read from a second file (containing change records rather than entries), and the updated entries will be written to a third LDIF file. Unlike ldapmodify, the ldifmodify cannot read the changes to apply from standard input.
ldifsearch	Search one or more LDIF files to identify entries matching a given set of criteria.

leave-lockdown-mode	Request that the Directory Proxy Server leave lockdown mode and resume normal operation.
list-backends	List the backends and base DNs configured in Directory Proxy Server.
load-ldap-schema-file	Load the schema definitions contained in a specified LDIF file into the schema for a running server. This tool can only be used in conjunction with a server instance running on the local system.
make-ldif	Generate LDIF data based on a definition in a template file. For example template files, see the server's `config/MakeLDIF` directory. In particular, the `examples-of-all-tags.template` file shows how to use all of the tags for generating values.
manage-account	Retrieve or update information about the current state of a user account. Processing is performed using the password policy state extended operation, and you must have the `password-reset` privilege to use this extended operation.
manage-certificates	Manage certificates and private keys in a JKS or PKCS #12 key store.
manage-extension	Install or update extension bundles. An extension bundle is a package of extension(s) that utilize the Server SDK to extend the functionality of the PingDirectoryProxy Server. Extension bundles are installed from a zip archive or file system directory. The Directory Proxy Server will be restarted if running to activate the extension(s).
manage-profile	Generate, compare, install, and replace server profiles.
manage-tasks	Access information about pending, running, and completed tasks scheduled in the Directory Proxy Server.
manage-topology	Manage the topology registry.
modrate	Perform repeated modifications against an LDAP directory server.
move-subtree	Move all entries in a specified subtree from one server to another.
parallel-update	Perform add, delete, modify, and modify DN operations concurrently using multiple threads.
prepare-external-server	Prepare and a directory server for communication..
profile-viewer	View information in data files captured by the Directory Proxy Server profiler.
register-yubikey-otp-device	Register a YubiKey OTP device with the Directory Server for a specified user so that the device may be used to authenticate that user with the UNBOUNDID-YUBIKEY-OTP SASL mechanism. Alternately, it may be used to deregister one or more YubiKey OTP devices for a user so that they may no longer be used to authenticate that user.
reload-http-connection-handler-certificates	Reload HTTPS Connection Handler certificates
reload-index	Reload the contents of the global index.
remove-attribute-type-from-schema	Safely remove an attribute type definition from the server schema.
remove-backup	Safely remove a backup and optionally all of its dependent backups from the specified Directory Proxy Server backend.
remove-defunct-server	Remove a server from this server's topology.

replace-certificate	Replace the listener certificate for this Directory Proxy Server server instance.
restore	Restore a backup of a Directory Proxy Server backend.
revert-update	Revert this server package's most recent update.
review-license	Review and/or indicate your acceptance of the license agreement defined in /legal/LICENSE.txt.
rotate-log	Trigger the rotation of one or more log files.
sanitize-log	Sanitize the contents of a server log file to remove potentially sensitive information while still attempting to retain enough information to make it useful for diagnosing problems or understanding load patterns. The sanitization process operates on fields that consist of name-value pairs. The field name is always preserved, but field values might be tokenized or redacted if they might include sensitive information. Supported log file types include the file-based access, error, sync, and resync logs, as well as the operation timing access log and the detailed HTTP operation log. Sanitize the audit log using the scramble-ldif tool.
schedule-exec-task	Schedule an exec task to run a specified command in the server. To run an exec task, a number of conditions must be satisfied: the server's global configuration must have been updated to include 'com.unboundid.directory.server.tasks.ExecTask' in the set of allowed-task values, the requester must have the 'exec-task' privilege, and the command to execute must be listed in the 'exec-command-whitelist.txt' file in the server's `config` directory. The absolute path (on the server system) of the command to execute must be specified as the first unnamed trailing argument to this program, and the arguments to provide to that command (if any) should be specified as the remaining trailing arguments. The server root is used as the command's working directory, so any arguments that represent relative paths are interpreted as relative to that directory.
search-and-mod-rate	Perform repeated searches against an LDAP directory server and modify each entry returned.
search-logs	Search across log files to extract lines matching the provided patterns, like the **`grep`** command-line tool. The benefits of using this tool over **`grep`** are its ability to handle multi line log messages, extract log messages within a given time range, and the inclusion of rotated log files.
searchrate	Perform repeated searches against an LDAP directory server.
server-state	View information about the current state of the Directory Proxy Server process.
set-delegated-admin-aci	Request that the Directory Proxy Server assign appropriate ACI for configured delegated administrators of the Delegated Admin API.
setup	Perform the initial setup for the Directory Proxy Server instance.
start-server	Start the Directory Proxy Server.
status	Display basic server information.
stop-server	Stop or restart the server.
subtree-accessibility	List or update the set of subtree accessibility restrictions defined in the Directory Proxy Server.
sum-file-sizes	Calculate the sum of the sizes for a set of files.

summarize-access-log	Generate a summary of one or more access logs to display a number of metrics about operations processed within the server.
transform-ldif	Apply one or more changes to entries or change records read from an LDIF file, writing the updating records to a new file. This tool can apply a variety of transformations, including scrambling attribute values, redacting attribute values, excluding attributes or entries, replacing existing attributes, adding new attributes, renaming attributes, and moving entries from one subtree to another.
uninstall	Uninstall Directory Proxy Server.
update	Update the Directory Proxy Server to a newer version by downloading and unzipping the new server install package on the same host as the server you wish to update. Then, use the **`update`** tool from the new server package to update the older version of the server. Before upgrading a server, you should ensure that it is capable of starting without severe or fatal errors. During the update process, the server is stopped if running, then the update performed, and a check is made to determine if the newly updated server starts without major errors. If it cannot start cleanly, the update will be backed out and the server returned to its prior state. See the **`revert-update`** tool for information on reverting an update.
validate-acis	Validate a set of access control definitions contained in an LDAP server (including Sun/Oracle DSEE instances) or an LDIF file to determine whether they are acceptable for use in the Directory Proxy Server. Note that output generated by this tool is LDIF, but each entry in the output will have exactly one ACI, so entries that have more than one ACI will appear multiple times in the output with different ACI values.
validate-file-signature	Validate file signatures. For best results, file signatures should be validated by the same instance used to generate the file. However, it might be possible to validate signatures generated on other instances in a replicated topology.
validate-ldap-schema	Validate an LDAP schema read from one or more LDIF files.
validate-ldif	Validate the contents of an LDIF file against the server schema.

Saving Options in a File

Creating a tools properties file

You can set properties that apply to all tools or are tool-specific. These properties serve as defaults for the command-line options they represent.

Steps

1. Use a text editor to open the default tools properties file (`config/tools.properties`) or a different properties file.

> ⓘ **Note:**
>
> If you use a file other than `config/tools.properties`, invoke the tool with the `--propertiesFilePath` option to specify the path to your properties file.

2. Set or change properties that apply to all tools.

   Use the standard Java properties file format (name=value) to set properties. For example, the following properties define a set of LDAP connection parameters.

   ```
   hostname=server1.example.com
   port=1389
   bindDN=cn=Directory\ Manager
   bindPassword=secret
   baseDN=dc=example,dc=com
   ```

   > ⓘ **Note:**
   >
   > Properties files do not allow quotation marks of any kind around values.
   >
   > Escape spaces and special characters.
   >
   > Whenever you specify a path, do not use ~ to refer to the home directory. The server does not expand the ~ value when read from a properties file.

3. Set or change properties that apply to specific tools.

   Tool-specific properties start with the name of the tool followed by a period. These properties take precedence over properties that apply to all tools. The following example sets two ports: one that applies to all tools (`port=1389`) and a tool-specific one that **`ldapsearch`** uses instead (`ldapsearch.port=2389`).

   ```
   hostname=server1.example.com
   port=1389
   ldapsearch.port=2389
   bindDN=cn=Directory\ Manager
   ```

4. Save your changes and close the file.

Evaluation order

Directory Proxy Server uses the following evaluation ordering to determine options for a given command-line tool:

- All options on the tool's command line take precedence over any options in any properties file.
- If you use the `--propertiesFilePath` option with no other options, Directory Proxy Server takes its options from the specified properties file.
- If you use no options on the command line including the `--propertiesFilePath` option (and `--noPropertiesFile`), Directory Proxy Server searches for the `tools.properties` file at *<server-root>*.
- If no default properties file is found and a required option is missing, the tool generates an error.
- Tool-specific properties (for example, `ldapsearch.port=3389`) take precedence over general properties (for example, `port=1389`).

Consider this example properties file that is saved as *<server-root>*`/bin/tools.properties`:

```
hostname=server1.example.com
port=1389
bindDN=cn=Directory\ Manager
bindPassword=secret
```

Directory Proxy Server locates a command-line option in a specific priority order.

1. All options presented with the tool on the command line take precedence over any options in any properties file. In the following example, the client request is run with the options specified on the

command line (`--port` and `--baseDN`). The command uses the `bindDN` and `bindPassword` values specified in the properties file.

```
$ bin/ldapsearch --port 2389 --baseDN ou=People,dc=example,dc=com \
  --propertiesFilePath bin/tools.properties "(objectclass=*)"
```

2. Next, if you specify the properties file using the `--propertiesFilePath` option and no other command-line options, Directory Proxy Server uses the specified properties file as follows:

```
$ bin/ldapsearch --propertiesFilePath bin/tools.properties \
  "(objectclass=*)"
```

3. If no options are presented with the tool on the command line and the `--noPropertiesFile` option is not present, Directory Proxy Server attempts to locate any default `tools.properties` file in the following location:

```
<server-root>/config/tools.properties
```

   If you move your `tools.properties` file from *<server-root>*`/bin` to the *<server-root>*`/config` directory. You can then run your tools as follows:

```
$ bin/ldapsearch "(objectclass=*)"
```

   You can Directory Proxy Server so that it does not search for a properties file by using the `--noPropertiesFile` option. This options tells Directory Proxy Server to use only those options specified on the command line. You cannot use the `--propertiesFilePath` and `--noPropertiesFile` options together.

4. If no default `tools.properties` file is found and no options are specified with the command-line tool, then the tool generates an error for any missing arguments.

Sample dsconfig batch files

The `config/sample-dsconfig-batch-files` directory contains **`dsconfig`** batch files that you can use to configure various aspects of the server. For example, these files can enable additional security capabilities or take advantage of features that might require customization from one environment to another.

Each file includes comments that describe the purpose and benefit of its configuration change. You can choose which of the changes you want to apply.

You need to customize some of the batch files to provide values that might vary from one environment to another. To apply a batch file that requires changes, copy it to another directory and edit the copy. Leave the files in the `config/sample-dsconfig-batch-files` directory unchanged so that they can be updated when you upgrade the server. To specify the path to the file that contains the changes to apply, use the **`dsconfig`** tool (**`bin/dsconfig`** on UNIX-based systems or **`bat\dsconfig.bat`** on Windows) with the `--batch-file` argument.

You should also provide the arguments needed to connect and authenticate to the server. The `--no-prompt` argument ensures that the tool does not block while waiting for input if any necessary arguments are missing. Consider this example.

```
bin/dsconfig --hostname localhost \
  --port 636 --useSSL --trustStorePath config/truststore \
  --bindDN "uid=admin,dc=example,dc=com" \
  --bindPasswordFile admin-password.txt \
  --batch-file config/hardening-dsconfig-batch-files/reject-insecure-
request.dsconfig \
  --no-prompt
```

Running task-based tools

Directory Proxy Server has a Tasks subsystem that allows you to schedule basic operations, such as **`backup`**, **`restore`**, **`rotate-log`**, **`schedule-exec-task`**, **`stop-server`**, and others. All task-based tools require the `--task` option that explicitly indicates the tool is to run as a task rather than in offline mode.

The following table shows the options that you can use for task-based operations:

Options for task-based operations

Option	Description
`--task`	Indicates that the tool is invoked as a task. The `--task` option is required. If you invoke a tool as a task without this `--task` option, then a warning message is displayed stating that it must be used. If the `--task` option is provided but the tool was not given the appropriate set of authentication arguments to the server, then an error message is displayed and the tool exits with an error.
`--start` *<startTime>*	Indicates the date and time, expressed in the format 'YYYYMMDDhhmmss', when the operation is to start. A value of '0' causes the task to be scheduled for immediate execution. After the scheduled run, the tool exits immediately.
`--dependency` *<taskID>*	Specifies the ID of a task upon which this task depends. A task does not start execution until all its dependencies have completed execution. You can use this option multiple times in a single command.
`--failedDependencyAction` *<action>*	Specifies the action this task takes if one of its dependent tasks fail. Valid action values are: ▪ `CANCEL` (the default) Cancels the task. ▪ `DISABLE` Disables the task so that it is not eligible to run until you manually enable it again. ▪ `PROCESS` Runs the task.
`--startAlert`	Generates an administrative alert when the task starts running.
`--errorAlert`	Generates an administrative alert when the task fails to complete successfully.
`--successAlert`	Generates an administrative alert when the task completes successfully.
`--startNotify` *<emailAddress>*	Specifies an email address to notify when the task starts running. You can use this option multiple times in a single command.

Option	Description
`--completionNotify <emailAddress>`	Specifies an email address to notify when the task completes, regardless of whether it succeeded or failed. You can use this option multiple times in a single command.
`--errorNotify <emailAddress>`	Specifies an email address to notify if an error occurs when this task executes. You can use this option multiple times in a single command.
`--successNotify <emailAddress>`	Specifies an email address to notify when this task completes successfully. You can use this option multiple times in a single command.

Consent Solution Guide

PingDirectory™ Product Documentation

Support

https://support.pingidentity.com/

Introduction to the Consent Service and Consent API

Companies gain loyalty and trust when they offer transparency and control to their users and customers regarding the personal data that is collected, processed, or shared. In Europe, the General Data Protection Regulation (GDPR) was designed specifically for allowing companies to collect and use valuable data about its users, while protecting the rights of citizens to control what is collected and used. To support the collection and end-user control of personal data, PingDirectory Server includes schema and REST APIs that provide the ability to collect fine-grained data authorizations (consents), from users and customers.

Consent Service overview

The Consent Service is an HTTP-based REST API hosted by the PingDirectory Server or PingDirectoryProxy Server. The service enables the collection of consent from application users, the

enforcement of consent, a user's management of his or her consent, and auditing of consent actions. Enterprises can integrate these features into their applications to give users transparency and control of their data privacy.

For the purpose of this document, the following terms are used:

Consent Terms

Term	Meaning
collaborators	A list of individuals for whom access to this consent record is shared.
Consent definition	The terms of the fine-grained contract, which describes the data that can be processed or shared, and a purpose for processing or sharing the data. The consent definition is stored in the server configuration.
Consent localization	A child object of a definition that contains versioned, localized text for the consent definition, to be used when prompting an individual. This is stored in the server configuration.
Consent record	A record of a consent interaction with a user. Consent records are stored in the directory tree.
Subject	The individual whose data can be collected, processed, or shared.
Actor	The individual who granted/denied/revoked consent. This is usually the same as the subject.
Audience	The entity, application, or service that is granted or denied access to a subject's data for a specific purpose.

Consent API overview

The PingDirectory Server and PingDirectoryProxy Server provide a REST API for managing individuals' consent to handle their data. This can be used as a component of a larger solution, such as a GDPR compliance system or the PingDataGovernance Server Open Banking Account Requests API.

The PingDirectory Server Consent API enables authorization services:

- to capture user consent for sharing or processing data
- to confirm that consent to share or process data has been granted
- for individuals to manage the consent that they have granted.

Detailed API documentation can be found on the Ping Identity website.

How consents are collected

User consent is collected by creating a consent record through the Consent API. In most cases, the Consent API client uses consent localization data to construct an approval prompt to display to the user. This prompt should include text describing what data is collected and for what purpose, allowing the user to make an informed decision about the value of sharing his or her data.

For example, a web application needing to collect consent for a user's browsing behavior would use the Consent API to look up the localizations for the **`browsing-behavior`** consent definition. It would select the localization appropriate for the user and use data from the localization resource to construct a consent prompt for display to the user. After the user is prompted and makes a decision, the client could store the decision by creating a new consent record through the Consent API.

How consents are enforced

The Consent Service can be used as a data source for making access control decisions. If a particular data usage scenario requires consent, then the application or service needing to access or process that

data must not be able to use the data unless the user has provided consent. The entity that performs this consent check may be the application itself or some other service.

To perform a consent check, the Consent API client must be able to correlate a data access request type with a consent definition. For example, if a web application needs to collect a user's browsing behavior, this data collection scenario might be represented by a consent definition called **`browsing-behavior`**. The application would check for an existing consent grant by searching the Consent API for a consent record that matches the user and the **`browsing-behavior`** consent definition. If a match is found, then the application can proceed. If a match is not found, the application must collect consent from the user.

How applications use the Consent API

The following example illustrates both consent capture and consent enforcement. This example follows a user's journey on a website during which the company must gather consent to track the user's browsing behavior:

1. A user launches the company's application and authenticates. The application wants to record the page visit, but first it must check if the user has granted consent to do so.
2. The application makes a call to the Consent API to determine if the **`browsing-behavior`** consent record exists for this user, and whether consent been granted.
3. The API returns a result indicating that no consent record exists. The application must prompt the user for his or her consent. The application calls the Consent API to retrieve the localization for the **`browsing-behavior`** consent, which includes the language that the application uses to produce a prompt for the user.
4. After the user makes a decision, the application stores the user's decision by creating a new consent record. This is through a call to the Consent API.
5. Later, the user visits another page in the company's site. The application wants to record the page visit, and again checks whether the user has granted consent to do so.
6. The application makes a call to the Consent API to get the **`browsing-behavior`** consent record for this user.
7. If the user's consent record agrees to have the company track his or her browsing behavior, the application can then make the appropriate calls to track browsing behavior. This is consent enforcement.

Configuring the Consent Service

This section provides details for installing and configuring the components on which the Consent Service relies. Refer to the PingDirectory Server Administration Guide for detailed configuration information.

Configuration overview

The Consent Service is not enabled by default. The setup and configuration process varies depending on the following factors:

- Whether client applications will allow an individual to self-manage consents.
- Whether some or all client applications will be privileged, with the ability to manage all consents.
- The HTTP authentication method used by client applications.
- Whether consent records exist in the same directory as user entries.

Example configuration scenarios

The following client application scenarios are available for determining how the Consent Service should be configured to meet your business needs.

Directly managed consents

In this scenario, one or more client applications provide an interface for individuals to directly manage their own consent records. These applications can only manage consents for the currently authenticated user. In addition, there is also a client application for consent administrators. An OAuth 2 authorization server grants access tokens that the applications uses to access the Consent API.

Configuration for this scenario includes:

1. Configure an OAuth 2 authorization server to issue a **`urn:pingdirectory:consent`** scope to individuals and a **`urn:pingdirectory:consent_admin`** scope to consent administrators.
2. Create an identity mapper to map subject identifiers used by the authorization server to LDAP DNs used by the PingDirectory Server.
3. Configre an access token validator to validate tokens issued by the OAuth 2 authorization server.
4. Configure the Consent HTTP Servlet Extension to disable HTTP basic authentication and restart the HTTPS Connection Handler.
5. Configure the Consent Service to use the OAuth scopes and token validator.

Indirectly managed consents (basic authentication)

In this scenario, an application uses a privileged service account to manage its users' consents. The application's privileged account can access any consent record, which gives the application the ability to perform operations that an individual user cannot. The following include steps the setup needed for the PingDataGovernance Server's Open Banking Account Requests service to use the Consent Service as its backend.

Configuration for this scenario includes:

1. Create a service account for the application.
2. Configure the Consent HTTP Servlet Extension to enable HTTP basic authentication and restart the HTTPS Connection Handler.
3. Create an identity mapper to map consent record subject and actor attribute values to LDAP DNs. This is optional.
4. Configure the Consent Service to use the application's service account, and optionally the identity mapper.

Setting up with the configuration scripts

PingDirectory Server includes two configuration scripts that can serve as the starting point for setting up the Consent Service. Both scripts must be carefully reviewed and updated to support your client application scenarios and business needs.

- **`consent-service-base-entries.ldif`** - This LDIF script can be imported to create the base DN where consent records will be stored.
- **`consent-service-cfg.dsconfig`** - This script can be imported to configure and enable the Consent Service.

Both are located in the **`/resource/consent/`** directory of the PingDirectory Server server root.

Basic configuration with the **`consent-service-base-entries.ldif`** file includes:

1. Edit the LDIF script and change the location of where consent records will be stored.
2. Import the LDIF script using the **`ldapmodify`**command, such as:

```
$ bin/ldapmodify --defaultAdd \
  --filename consent-service-base-entries.ldif
```

Basic configuration with the **`consent-service-cfg.dsconfig`** file includes:

1. Search for **`CHANGE-ME`** and replace values.

2. Review configuration commands and make additional changes to match existing Ping environment parameters, application scenarios, and business needs.
3. Impost the script with the **`dsconfig`** command, such as:

```
$ bin/dsconfig --no-prompt \
  --batch-file consent-service-cfg.dsconfig
```

Setting up in a replicated PingDirectory Server environment

Running the Consent Service setup script requires special consideration in an environment that includes replicated PingDirectory Servers. If possible, setup the Consent Service after replication is enabled for the PingDirectory Servers. See the *PingDirectory Server Administration Guide* for details about server replication.

Set up Consent Service after replication is enabled

Complete the following steps if replication is already enabled for PingDirectory Servers.

1. If needed, configure the PingDirectory Servers to use a configuration group called "all-servers." This will ensure that configuration changes are applied to all servers in a topology.

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

2. Run the Consent Service setup script.

```
$ bin/dsconfig --no-prompt \
  --batch-file resource/consent/consent-service-cfg.dsconfig
  --applyChangeTo server-group
```

Set up Consent Service before replication is enabled

If you have already set up the Consent Service on a standalone PingDirectory Server, perform the following the steps before enabling replication. In this example, "DS1" is the original PingDirectory Server, and "DS2" is the second server that will be added as a replica.

1. Run the **`config-diff`** command without arguments on DS1 to produce a batch file that contains configuration changes that will be applied to DS2.

```
$ bin/config-diff > config-changes.dsconfig
```

2. Apply the **`config-changes.dsconfig`** file to DS2.

```
$ bin/dsconfig --no-prompt \
  --batch-file config-changes.dsconfig \
  --applyChangeTo single-server
```

3. Restart DS2.
4. Enable replication between the two servers.

Configuration reference

There are many configuration options for the Consent Sevice and application integration. The configuration scripts included with the PingDirectory Server provide a starting point. Additional detailed information about the Consent Service properties and configuration is provided as reference.

General Consent Service configuration

The Consent Service configuration is used to control authorization behavior and determines where consent records are stored in the PingDirectory Server. The service properties are configured with the **dsconfig set-consent-service-prop** command. The consent service configuration script configures the consent service properties as follows:

```
$ bin/dsconfig set-consent-service-prop \
  --set enabled:true \
  --set base-dn:ou=consents,dc=example,dc=com \
  --set "bind-dn:cn=consent service account" \
  --set unprivileged-consent-scope:urn:pingdirectory:consent \
  --set privileged-consent-scope:urn:pingdirectory:consent_admin \
  --set "consent-record-identity-mapper:User ID Identity Mapper"
```

The following are Consent Service properties.

Consent Service properties

Property	Description	Required to enable service
enabled	If set to true, enables the Consent Service for handling client requests.	Yes
base-dn	Specifies a container DN for consent record entries.	Yes
bind-dn	Specifies an internal service account used by the Consent Service to perform LDAP operations.	Yes
service-account-dn	Specifies one or more DNs of requesters that will be considered privileged when using basic authentication. If not defined, a requester will only be considered privileged if it is mapped to a DN with the **bypass-acl** privilege. Optional.	No
unprivileged-consent-scope	Specifies the name of the scope required for bearer tokens representing unprivileged requesters.	Yes
privileged-consent-scope	Specifies the name of the scope required for bearer tokens representing privileged requesters.	Yes
consent-record-identity-mapper	Specifies one or more identity mappers used to map consent record **subject** and **actor** values to DNs. By default, these values are inferred from the authentication context, such as the bearer token subject. Optional.	No
audience	Specifies an **audience** claim value that the Consent Service will require to be present in bearer tokens that it accepts. Optional.	No

For the Consent Service to report itself as available to clients, the following must be true:

- The Consent Service must be enabled.
- The Consent Service base DN must be configured and must exist.
- The internal service account must be configured and must exist.
- The internal service account must have the right to read, add, modify, and delete entries under the Consent Service base DN.

Creating a container entry for consent records

About this task

Each consent record is a distinct entry in the PingDirectory Server, and the Consent Service requires that these entries be stored under a common base DN, defined by the **`base-dn`** property of the Consent Service configuration. The Consent Service LDIF file sets the base DN. Use these steps to choose a different location to store consent records.

Steps

1. To create the Consent Service base DN, open a text editor and save the following to the file **`consent-service-base-dn.ldif`**.

```
dn: ou=consents,dc=example,dc=com
objectClass: top
objectClass: organizationalUnit
ou: consents
```

2. Use **`ldapmodify`** to add the entry.

```
$ bin/ldapmodify --defaultAdd --filename consent-service-base-dn.ldif
```

Creating an internal service account

About this task

The Consent Service uses an internal LDAP connection to operate against consent records that are stored as LDAP entries. It authenticates this LDAP connection using a service account, which must be created and dedicated solely to the Consent Service.

The Consent Service configuration script configures the internal service account using a topology admin user. If needed, this can be changed to a root DN user or a user DN whose entry is in the user backend. In all cases, the service account should exist in every LDAP server in the topology.

This service account must have full read and write access to the Consent Service base DN, the ability to read users' **`isMemberOf`** attribute, and the right to use the following LDAP controls:

- IntermediateClientRequestControl (1.3.6.1.4.1.30221.2.5.2)
- NameWithEntryUUIDRequestControl (1.3.6.1.4.1.30221.2.5.44)
- RejectUnindexedSearchRequestControl (1.3.6.1.4.1.30221.2.5.54)
- PermissiveModifyRequestControl (1.2.840.113556.1.4.1413)
- PostReadRequestControl (1.3.6.1.1.13.2)

For more information about configuring access, see the *"Managing Access Control"* chapter of the PingDirectory Server Administration Guide.

Steps

- To ensure the correct access, create a user with the **`bypass-acl`** privilege. The following **`dsconfig`** command creates a topology admin user with the **`bypass-acl`** privilege. After this is created, set this user as the **`bind-dn`** for the Consent Service.

```
$ dsconfig create-topology-admin-user \
  --user-name "Consent Service Account" \
  --set "description:Consent API service account" \
  --set "alternate-bind-dn:cn=consent service account" \
  --set first-name:Consent \
  --set inherit-default-root-privileges:false \
  --set last-name:Service \
  --set password:CHANGE-ME \
  --set privilege:bypass-acl
```

- Because the **bypass-acl** privilege grants a broad level of access, you may not want to grant this privilege to the Consent Service account. If desired, add the following ACI to enable a targetted set of functionality for the Consent Service. The following example grants this access to the DN **cn=consent service account** using global ACIs:

```
# Grant access to the consent record base DN ou=consents,dc=example,dc=com
dsconfig set-access-control-handler-prop --add 'global-aci:(target="ldap:///
ou=consents,dc=example,dc=com")(targetattr="*||+")(version 3.0; acl "Consent
 Service account access to consent record data"; allow(all) userdn="ldap:///
cn=consent service account";)'

# Grant access to the LDAP request controls used by the Consent Service.
dsconfig set-access-control-handler-prop --add 'global-aci:
(targetcontrol="1.3.6.1.4.1.30221.2.5.2||1.3.6.1.4.1.30221.2.5.44||
1.3.6.1.4.1.30221.2.5.54||1.2.840.113556.1.4.1413||1.3.6.1.1.13.2")(version
 3.0; acl "Consent Service account access to selected controls"; allow
 (read) userdn="ldap:///cn=consent service account";)'
```

Configure an identity mapper

The Consent Service uses identity mappers to map requester identities, subject values, and actor values to DNs. An identity mapper takes a user identifier string and correlates the identifier with the DN of a user entry. The PingDirectory Server provides four different types of identity mappers.

Identity mappers

Identity mapper type	Description
Exact match identity mapper	Maps a user identifier to a DN by searching for an entry with an attribute that exactly matches the identifier.
Regular expression identity mapper	Similar to an exact match identity mapper, but allows a regular expression to be specified for more flexible matching.
Third-party identity mapper	A custom Java identity mapper implementation written using the Server SDK.
Groovy scripted identity mapper	A custom Groovy identity mapper implementation written using the Server SDK.

The Consent Service can be configured to use identity mappers for each of the following scenarios:

- Requesters authenticating using basic authentication - use the Consent HTTP Servlet Extension **identity-mapper** property to configure an identity mapper that takes the HTTP Basic authorization user name string to find the corresponding user's identity in the PingDirectory Server.
- Requesters authenticating using bearer token authentication - use the Access Token Validator **identity-mapper** property to configure an identity mapper that takes the subject (or other claim value from the OAuth token) to find the corresponding user's identity in the PingDirectory Server.
- Consent record actor and subject values - use the Consent Service **consent-record-identity-mapper** property to configure an identity mapper that takes these consent record attribute values and uses them to find the corresponding users' identities in the PingDirectory Server.

The consent record identity mapper

By default, the Consent Service automatically sets the subject, subjectDN, actor, and actorDN values to the identity of the authenticated requester. If the requester uses basic authentication, then all values will be set to the auth DN determined by the basic authentication identity mapper. If the requester uses bearer token authentication, then the subject and actor values are set to the bearer token's subject claim value, while the subjectDN and actorDN values will be set to the auth DN determined by the access token validator identity mapper.

Privileged clients may manually set a consent record's subject and/or actor values. In those cases, the Consent Service's **consent-record-identity-mapper** property is used to map a consent record's subject and/or actor values to subjectDN and actorDN values, respectively.

Identity mapper configuration options

The Consent Service configuration script configures a single identity mapper to be used for all three scenarios. The provided identity mapper searches by **uid**, **cn**, or **entryUUID** attributes under the base DNs **cn=config and ou=people,dc=example,dc=com**.

The following configuration provides an example of an identity mapper that will match a user identifier to an LDAP entry with the same value in its **uid** attribute:

```
$ bin/dsconfig create-identity-mapper --mapper-name "User ID Exact Match" \
  --type exact-match \
  --set enabled:true \
  --set match-attribute:uid
```

The following configuration shows another typical example, that of an identity mapper that will match a user identifier to an LDAP entry with the same value in its **entryUUID** attribute:

```
$ bin/dsconfig create-identity-mapper --mapper-name "EntryUUID Exact Match"
 \
  --type exact-match \
  --set enabled:true \
  --set match-attribute:entryUUID
```

The last example creates an identity mapper that will match a user identifier to an LDAP entry with the same value in either its **uid**, **cn**, or **entryUUID** attribute. This identity mapper will also constrain its search to the **ou=people,dc=example,dc=com** and **cn=config** base DNs. (The **cn=config** base DN is not searched by default, and must be explicitly listed to be searched.)

```
$ bin/dsconfig create-identity-mapper \
  --mapper-name "User ID Identity Mapper" \
  --type exact-match \
  --set enabled:true \
  --set match-attribute:uid \
  --set match-attribute:cn \
  --set match-attribute:entryUUID \
  --set match-base-dn:cn=config \
  --set match-base-dn:ou=people,dc=example,dc=com
```

Authentication methods

The Consent Service supports two HTTP authentication methods, which are both enabled by default:

- Basic authentication
- Bearer token authentication

The Consent servlet looks at the request's Authorization header to determine which authentication type is being used by the client.

With basic authentication, the client provides an encoded user name-password pair in the HTTP Authorization request header. When the Consent Service receives a request using basic authentication, it maps the user name credential to a DN using an identity mapper. This DN is designated the **auth DN** and is used to make subsequent authorization decisions. The Consent Service then performs an LDAP bind using the DN and password to determine if the request can be processed.

With bearer token authentication, the client provides an access token in the HTTP Authorization request header. The access token is always obtained by the client from an external OAuth 2 authorization server

and encapsulates information ("claims") about a user identity, the client identity, and the requests that the client is authorized to make.

The PingDirectory Server must be configured to accept access tokens using one or both available access token validators:

- **PingFederate access token validator**. Supports access tokens issued by a PingFederate authorization server. This validator verifies an access token and discovers its claims by making a request to the PingFederate server's token introspection endpoint.

  > ⓘ **Note:**
  >
  > If you are using PingFederate 10.0 or earlier, ensure that PingFederate is configured to respond to OAuth and OpenID Connect (OIDC) requests by selecting the **Enable OAuth 2.0 Authorization Server (AS) role** and **OpenID Connect** check boxes as explained in *Enabling the OAuth AS role*. Starting with PingFederate 10.1, these items are always enabled.

- **JWT access token validator**. Supports signed or encrypted JWT access tokens issued by an arbitrary authorization server. This validator checks an access token by cryptographically verifying the token's signature using a trusted public certificate. The token's claims are encoded in the token itself, so discovering the token's claims does not require an outgoing token introspection request.

The token validator uses its identity mapper to map the subject claim to a DN. This DN is designated the **`auth DN`** and is used along with the token's claims to make subsequent authorization decisions.

If the PingDirectory Server is configured with at least one access token validator, it will be used by the Consent Service. If the PingDirectory Server is configured with more than one access token validator, the validators are consulted in order until one is able to successfully authenticate the request.

If the PingDirectory Server is configured with multiple access token validators, but only one should be used by the Consent Service, the access token validator can be configured by setting the **`access-token-validator`** property of the Consent HTTP Servlet Extension.

> ⓘ **Note:** Configuring an access token validator for the Consent Service requires information from the authorization server configuration:
>
> - The values that the authorization server sets for **`subject`** claims must be mappable to a DN in the PingDirectory Server.
> - The authorization server must be configured to authorize clients and grant scopes appropriately for privileged or unprivileged Consent API access.
> - The authorization server must be configured to issue tokens with scopes corresponding to the Consent Service's **`unprivileged-scope-name`** and **`privileged-scope-name`** configuration.
>
> Refer to the authorization server's documentation for guidance.

Configuring basic authentication

About this task

Basic authentication is enabled by default, and the settings are configured in the Consent HTTP Servlet Extension configuration.

Steps

- Use the following command to disable basic authentication.

```
$ bin/dsconfig set-http-servlet-extension-prop \
  --extension-name Consent \
  --set basic-auth-enabled:false
```

- Use the following command to enable basic authentication.

```
$ bin/dsconfig set-http-servlet-extension-prop \
  --extension-name Consent \
  --set basic-auth-enabled:true
```

- Use the following command to configure an identity mapper for basic authentication.

```
$ bin/dsconfig set-http-servlet-extension-prop \
  --extension-name Consent \
  --set "identity-mapper:User ID Exact Match"
```

- All of these configuration changes require the Consent servlet to be reloaded before they can take effect. Use the following commands to restart the connection handler that hosts the Consent servlet.

```
$ bin/dsconfig set-connection-handler-prop \
  --handler-name "HTTPS Connection Handler" \
  --set enabled:false
```

```
$ bin/dsconfig set-connection-handler-prop \
  --handler-name "HTTPS Connection Handler" \
  --set enabled:true
```

Configuring bearer token authentication

Steps

- The following is an example access token validator configured on the PingDirectory Server for a PingFederate server:

```
$ bin/dsconfig create-external-server \
  --server-name PingFederate \
  --type http \
  --set base-url:https://my-ping-federate-server:1443/
```

```
$ bin/dsconfig create-access-token-validator \
  --validator-name "PingFederate Token Validator" \
  --type ping-federate \
  --set enabled:true \
  --set "identity-mapper:User ID Exact Match" \
  --set authorization-server:PingFederate \
  --set client-id:id \
  --set client-secret:secret
```

- If more than one access token validator is configured on the PingDirectory Server, the Consent Service can be configured to use a single validator with the following command:

```
$ bin/dsconfig set-http-servlet-extension-prop \
  --extension-name Consent \
  --set "access-token-validator:PingFederate Token Validator"
```

Configuring Consent Service scopes

About this task

The Consent Service checks access tokens for a **`subject`** claim and uses an identity mapper to map the value to a DN, called the request DN or auth DN. If no request DN can be mapped, the request is rejected. In addition, the Consent Service will only accept an access token with a scope that it is configured to recognize.

- An unprivileged consent scope designates the requester as unprivileged. The scope's name is configured with the Consent Service's **`unprivileged-consent-scope`** property.
- A privileged consent scope designates the requester as privileged. This is configured using the Consent Service's **`privileged-consent-scope`** property.

The authorization server must also be configured to issue tokens with these scopes.

Steps

- The following example configures these scopes for the Consent Service.

```
$ bin/dsconfig set-consent-service-prop \
  --set unprivileged-consent-scope:consent \
  --set privileged-consent-scope:consent_admin
```

Authorization

The Consent Service's distinction between privileged and unprivileged requesters determines the type of operations that can be performed by requesters. During the authorization phase, the Consent servlet performs checks on both the bearer token claims (if present) and the **`auth DN`** to determine if the requester is privileged or unprivileged. These are summarized in the following table.

Available operations per requester type

Requester type	Description	Access determined by	Can create consent records	Can update consent records	Can delete consent records
Unprivileged	Requesters with no authority to operate on consent records other than their own.	A requester is considered unprivileged if it does not meet any of the criteria for a privileged requester. If using bearer token authentication, the access token must include a scope named by the **`unprivileged-consent-scope`** property of the Consent Service configuration. Also, an unprivileged requester can only perform actions on consent records where the subject DN matches the requester DN.	Yes. The subject/subjectDN and actor/actorDN values will be set based on the requester.	Yes, if the requester DN matches the subject DN.	No.

Requester type	Description	Access determined by	Can create consent records	Can update consent records	Can delete consent records
Privileged	A requester with the authority to perform any operation on any consent record.	When using basic authentication, a requester is considered privileged if the requester DN either has the **`bypass-acl`** privilege or is listed in the **`service-account-dn`** property of the Consent Service configuration. If using bearer token authentication, the access token must include a scope named by the **`privileged-consent-scope`** property of the Consent Service configuration.	Yes.	Yes.	Yes.

Bearer token check

If a bearer token was used, the following checks are performed:

- If the Consent Service's **`audience`** property is configured, the bearer token's audience claim must match the configured value.
- If the bearer token contains a scope matching the Consent Service's **`privileged-scope-name`** property, then the requester is considered privileged.
- If not, the bearer token must have a scope matching the Consent Service's **`unprivileged-scope-name`** property, and the requester is considered unprivileged.

Basic authentication check

If basic authentication is used, the following checks are performed:

- If the **`auth DN`** has the LDAP privilege **`bypass-acl`**, the requester is privileged.
- If the **`auth DN`** is listed in the Consent Service's **`service-account-dn`** property, the requester is privileged.
- If not, the requester is considered unprivileged.

Managing Consents

This section describes the tasks required to support the collection and end-user control of personal data, and manage users' consents.

Overview of consent management

The full lifecycle of consent management goes beyond collecting the user's consent. First, the terms of each consent contract must be centrally managed. After collecting consent, the user will want to review previously granted consents and potentially revoke some. Finally, companies will need to be able to trace the history of updates to any consent in order to resolve a dispute or respond to audit.

Consent definitions and localizations

Companies will want to centrally manage the language used when prompting a user to give consent. This is key to ensuring a consistent user experience across multiple applications, such as mobile and web. The Consent Service requires one or more consent definitions to be defined in the PingDirectory Server configuration. Each consent definition represents the combination of:

- The data to be collected or shared.
- The purpose for collecting or sharing this data.

For example, a consent definition could represent user email addresses, used to deliver a third party's email newsletter. A consent definition could also represent access to a user's network-connected IoT device, which would be used for a home automation task controlled by a third party.

Each consent definition must have one or more localization. A localization is a versioned object consisting of the data that a Consent API client needs to prompt a user for consent. When a consent record is accepted or denied by a Consent Service client, it must include a reference to a consent definition, locale, and version.

Creating consent definition and localization

Steps

- The following creates a consent definition and a localization for it.

```
$ bin/dsconfig create-consent-definition \
  --definition-name email_newsletter \
  --set "display-name:Email newsletter"
```

```
$ bin/dsconfig create-consent-definition-localization \
  --definition-name email_newsletter \
  --localization-name en-US \
  --set version:1.0 \
  --set "data-text:Your email address" \
  --set "purpose-text:To receive newsletter updates"
```

- The following example updates a localization and its version.

```
$ bin/dsconfig set-consent-definition-localization-prop \
  --definition-name email_newsletter \
  --localization-name en-US \
  --set version:1.1 \
  --set "data-text:Your preferred email address"
```

Perform an audit on consents

Changes to Consent Service resources are tracked by one of two types of audit logs. For examples of configuring either type of log, see the **`<server-root>/resource/consent-service-cfg.dsconfig`** script bundled with the server or *Logging*. This example uses the Consent Trace Logger. It represents Consent Service change events using the same field names used by the Consent API.

Log Publishers

Log publisher	Log publisher type	Description
collaborators	Trace logger key	The collaborators value, available only when the resource type is consent.
Consent Trace Logger	file-based-trace	Records Consent Service events at the Consent API level. Change events are recorded using messages of type **`audit`**.

Log publisher	Log publisher type	Description
Consent LDAP Audit Logger	file-based-audit	Records data changes at the LDAP level. In combination with a Request Criteria configuration object, an LDAP audit logger can be configured to record changes to Consent Service resources only.

Trace logger keys for auditing

Trace logger audit messages consist of a timestamp, the message type (**`CONSENT AUDIT`**), and a set of key/value pairs. A subset of important keys are described in the following table.

ⓘ **Note:** The keys used in trace log audit messages vary depending on the type of resource.

Log Publishers

Trace logger key	Description
`requestID`	A server-specific HTTP request ID. This value can be correlated with messages produced by other loggers.
`resourceType`	The type of Consent Service resource that was changed. Possible values are **`definition`**, **`localization`**, or **`consent`**.
`changeType`	The type of change recorded by this message. Possible values are **`create`**, **`update`**, or **`delete`**.
`attrsAdded`	A comma-delimited list of the attributes that were added to the resource.
`attrsUpdated`	A comma-delimited list of the attributes that were modified on the resource.
`attrsDeleted`	A comma-delimited list of the attributes that were removed from the resource.
`requestDN`	The DN of the requester, which is available only when the resource type is **`consent`**.
`definitionID`	The consent definition ID. If the resource type is **`definition`**, this identifies the definition that was changed. If the resource type is **`localization`**, this identifies the parent definition. If the resource type is **`consent`**, this identifies the consent record's related definition.
`locale`	The locale. If the resource type is **`localization`**, this identifies the localization (in combination with the definition ID). If the resource type is **`consent`**, this identifies the related localization (combined with the definition ID).
`consentID`	The consent record ID, available only when the resource type is **`consent`**.
`subject`	The subject value, available only when the resource type is **`consent`**.
`subjectDN`	The subject's mapped LDAP DN, available only when the resource type is **`consent`**.
`actor`	The actor value, available only when the resource type is **`consent`**.
`actorDN`	The actor's mapped LDAP DN, available only when the resource type is **`consent`**.

Trace logger key	Description
audience	The audience value, available only when the resource type is **consent**.
status	The consent status. Possible values are **pending**, **accepted**, **denied**, **revoked**, and **restricted**. Only available when the resource type is **consent**.
previousStatus	The previous consent status, if applicable. Only available when the resource type is **consent**.
msg	A multiline value that includes the complete body of the changed resource. If the action is an update or a delete, the resource's body before the change will be included.

Perform an audit

Consent resource changes for particular entities (such as a specific user, or a specific consent definition) can be audited by searching the trace log using a combination of one of the message keys and the desired value. For example, if an individual's LDAP DN is known, then the **subjectDN** key can be used to construct a text search for any audit log messages containing that DN. Any matching log messages would constitute a history of that individual's consent activity.

Example new consent record

The following is a sample record. this audit log message provides important values in a parseable key/value format, but also includes the entirety of the new consent record.

```
[22/May/2018:18:02:42.584 -0500] CONSENT AUDIT requestID=57
 requestDN="uid=user.0,ou=people,
  dc=example,dc=com" consentID="6cff325b-e092-4094-b7f9-5a30864b0d24"
 subject="user.0" subjectDN="uid=user.0,
  ou=People,dc=example,dc=com" actor="user.0"
 actorDN="uid=user.0,ou=People,dc=example,dc=com" audience="client1"
  definitionID="cats" locale="en-US" status="accepted"
 attrsAdded="actor,audience,createdDate,dataText,subject,
  purposeText,definition,id,updatedDate,actorDN,status,subjectDN"
 changeType="create" resourceType="consent" msg="
New Consent Record:
    {'id':'6cff325b-e092-4094-
b7f9-5a30864b0d24','status':'accepted','subject':'user.0','subjectDN':'uid=user.0,

 ou=People,dc=example,dc=com','actor':'user.0','actorDN':'uid=user.0,ou=People,dc=exampl
  'client1','definition':{'id':'cats','version':'1.0','locale':'en-
US'},'dataText':'Collect data about your
  cats','purposeText':'To recommend cat food flavors that will satisfy and
 delight your feline companion',

 'createdDate':'2018-05-22T23:02:42.553Z','updatedDate':'2018-05-22T23:02:42.553Z'}"
```

Example updated consent record

This example shows the complete consent record before and after it was updated. With the **attrsUpdated**, **status**, and **previousStatus** keys, one can determine that the **status** changed from accepted to revoked.

```
[22/May/2018:18:05:08.660 -0500] CONSENT AUDIT requestID=59
 requestDN="uid=user.0,ou=people,
   dc=example,dc=com" consentID="6cff325b-e092-4094-b7f9-5a30864b0d24"
 subject="user.0" subjectDN="uid=user.0,
```

```
   ou=People,dc=example,dc=com" actor="user.0"
 actorDN="uid=user.0,ou=People,dc=example,dc=com"
   audience="client1" definitionID="cats" locale="en-US" status="revoked"
 previousStatus="accepted"
   attrsUpdated="status" changeType="update" resourceType="consent" msg="
Previous Consent Record:
    {'id':'6cff325b-e092-4094-
b7f9-5a30864b0d24','status':'accepted','subject':'user.0','subjectDN':'uid=user.0,

 ou=People,dc=example,dc=com','actor':'user.0','actorDN':'uid=user.0,ou=People,dc=exampl
  'audience':'client1','definition':
{'id':'cats','version':'1.0','locale':'en-US'},'dataText':'Collect
  data about your cats','purposeText':'To recommend cat food flavors that
 will satisfy and delight your
  feline
 companion','createdDate':'2018-05-22T23:02:42.553Z','updatedDate':'2018-05-22T23:02:42
Updated Consent Record:
    {'id':'6cff325b-e092-4094-
b7f9-5a30864b0d24','status':'revoked','subject':'user.0','subjectDN':

 'uid=user.0,ou=People,dc=example,dc=com','actor':'user.0','actorDN':'uid=user.0,ou=Peop
  dc=com','audience':'client1','definition':
{'id':'cats','version':'1.0','locale':'en-US'},'dataText':
  'Collect data about your cats','purposeText':'To recommend cat food
 flavors that will satisfy and
  delight your feline
 companion','createdDate':'2018-05-22T23:02:42.553Z','updatedDate':'2018-05-22T23:05:08
```

Example deleted consent record

This example shows that a consent record has been deleted, and the complete representation of the consent record prior to its deletion is provided.

```
[22/May/2018:18:06:35.071 -0500] CONSENT AUDIT requestID=61
 requestDN="cn=directory manager"
   consentID="6cff325b-e092-4094-b7f9-5a30864b0d24" subject="user.0"
 subjectDN="uid=user.0,ou=People,
   dc=example,dc=com" actor="user.0"
 actorDN="uid=user.0,ou=People,dc=example,dc=com" audience="client1"
   definitionID="cats" locale="en-US" status="revoked"
 previousStatus="revoked" attrsDeleted="actor,audience,

 createdDate,dataText,subject,purposeText,definition,id,updatedDate,actorDN,status,subje
 changeType="delete"
   resourceType="consent" msg="
Deleted Consent Record:
    {'id':'6cff325b-e092-4094-
b7f9-5a30864b0d24','status':'revoked','subject':'user.0','subjectDN':

 'uid=user.0,ou=People,dc=example,dc=com','actor':'user.0','actorDN':'uid=user.0,ou=Peop
   dc=example,dc=com','audience':'client1','definition':
{'id':'cats','version':'1.0','currentVersion':
   '1.0','locale':'en-US'},'dataText':'Collect data about your
 cats','purposeText':'To recommend cat food
   flavors that will satisfy and delight your feline
 companion','createdDate':'2018-05-22T23:02:42.553Z',
   'updatedDate':'2018-05-22T23:05:08.655Z'}"
```

Logging

About this task

The PingDirectory Server trace log publisher is used for logging events generated by HTTP service operations. The trace logger can be used to observe, debug, and audit consent requests.

ⓘ **Note:** To create a log of consent audit events only, remove all message types except for **`consent-message-type:audit`**.

Steps

- The following example of creates a trace logger for all consent events, plus summaries of HTTP requests and responses.

```
$ bin/dsconfig create-log-publisher \
  --publisher-name "Consent Trace Logger" \
  --type file-based-trace \
  --set "description:Records Consent API operations" \
  --set enabled:true \
  --set consent-message-type:audit \
  --set consent-message-type:consent-created \
  --set consent-message-type:consent-deleted \
  --set consent-message-type:consent-retrieved \
  --set consent-message-type:consent-search \
  --set consent-message-type:consent-updated \
  --set consent-message-type:definition-created \
  --set consent-message-type:definition-deleted \
  --set consent-message-type:definition-retrieved \
  --set consent-message-type:definition-search \
  --set consent-message-type:definition-updated \
  --set consent-message-type:error \
  --set consent-message-type:localization-created \
  --set consent-message-type:localization-deleted \
  --set consent-message-type:localization-retrieved \
  --set consent-message-type:localization-search \
  --set consent-message-type:localization-updated \
  --set http-message-type:request \
  --set http-message-type:response \
  --set 'exclude-path-pattern:/**/*.css' \
  --set 'exclude-path-pattern:/**/*.eot' \
  --set 'exclude-path-pattern:/**/*.gif' \
  --set 'exclude-path-pattern:/**/*.ico' \
  --set 'exclude-path-pattern:/**/*.jpg' \
  --set 'exclude-path-pattern:/**/*.js' \
  --set 'exclude-path-pattern:/**/*.png' \
  --set 'exclude-path-pattern:/**/*.svg' \
  --set 'exclude-path-pattern:/**/*.ttf' \
  --set 'exclude-path-pattern:/**/*.woff' \
  --set 'exclude-path-pattern:/**/*.woff2' \
  --set 'exclude-path-pattern:/console/**' \
  --set 'exclude-path-pattern:/console/**/template/**' \
  --set log-file:logs/consent-trace \
  --set "retention-policy:File Count Retention Policy" \
  --set "retention-policy:Free Disk Space Retention Policy" \
  --set "rotation-policy:24 Hours Time Limit Rotation Policy" \
  --set "rotation-policy:Size Limit Rotation Policy"
```

Correlating user and consent data

In some cases, the organization that has been granted consent by a group of users may need to perform an LDAP search so that they can act upon consent data in the aggregate. For example, a marketing group has collected consent to send a newsletter by email. A search must be performed that will list all of the consent records where the consent definition is **`email`** and the status is **`accepted`**. Those records must be correlated to user entries, and each user's email address must be retrieved.

This task is performed with an LDAP search on the PingDirectory Server. Every consent record has a **`subject`**, the user whose data is collected and stored. The Consent Service can be configured so that it stores the subject's DN in the **`subjectDN`** field.

In the LDAP schema:

- A consent record's **`subjectDN`** field is the **`ping-consent-subject-dn`** attribute.
- A consent record's status is the **`ping-consent-state`** attribute.
- A consent record's definition ID is in the **`ping-consent-definition.id`** JSON attribute field.
- And a user entry's email address is in the **`mail`** attribute.

The search will need to find all of the consent record entries where **`ping-consent-definition.id`** is **`email`** and the **`ping-consent-status`** is **`accepted`**. It then needs to correlate those consent record entries to user entries using **`ping-consent-subject-dn`**, and retrieve each user entry's **`mail`** attribute value. For example:

```
$ bin/ldapsearch \
   --baseDN "ou=consents,dc=example,dc=com" \
   --searchScope sub \
   --joinRule "dn:ping-consent-subject-dn" \
   --joinBaseDN "ou=people,dc=example,dc=com" \
   --joinScope sub \
   --joinRequestedAttribute mail
   '&(ping-consent-
definition:jsonObjectFilterExtensibleMatch:={ "filterType" : "equals",
 "field" : "id", "value" : "email" })(ping-consent-state=accepted)' \
   1.1

   # Join Result Control:
   #      OID:  1.3.6.1.4.1.30221.2.5.9
   #      Join Result Code:  0 (success)
   #      Joined With Entry:
   #           dn: uid=user.0,ou=People,dc=example,dc=com
   #           mail: user.0@example.com
   dn: entryUUID=9e481010-8330-425a-
bbf1-6637de053d48,ou=Consents,dc=example,dc=com

   # Result Code:  0 (success)
   # Number of Entries Returned:  1
```

The output listed under "Join Result Control" specifies the mail value.

Troubleshooting

The following are general guidelines for troubleshooting the Consent Service and any connection issues. When evaluating the configuration, make sure these issues are addressed first:

- Is the Consent Service enabled?
- Does the Consent Service base DN exist?
- Does the Consent Service's service account have the correct permissions?

- If the Consent Service should accept bearer tokens:
  - Are one or more Access Token Validators correctly configured?
  - Are the identity mappers for the Access Token Validators configured correctly?
  - Are the authorization servers correctly configured to issue tokens that the Consent Service will accept? Check the **`audience`**, **`privileged-consent-scope`**, and **`unprivileged-consent-scope`** properties of the Consent Service configuration.
- If privileged users are defined, are the members of the LDAP group specified by the Consent Service configuration's **`privileged-users-group-dn`** property?
- If there are applications that allow individuals to manage their own consents, is the system properly configured to map **`actor`** and **`subject`** DNs? Check the Consent Service configuration's **`consent-record-identity-mapper`** property.

Error cases

Consent Service is unavailable

If the Consent Service is unavailable, check that the service is enabled and that the communcation with the service is available. Confirm that the service account for the Consent Service has been properly provisioned. If the Consent Service resides on a PingDirectoryProxy Server, make sure that the service account exists on the PingDirectoryProxy Server and all PingDirectory Server behind the PingDirectoryProxy Server.

Requester lacks sufficient rights to perform operation

A request may be rejected with a 403 for the following reasons:

- The bearer token does not contain a required scope. Check the **`privileged-consent-scope`** and **`unprivileged-consent-scope`** properties of the Consent Service configuration.
- The bearer token does not contain a required **`audience`** claim. Check the **`audience`** property of the Consent Service configuration.
- Authentication was successful, but the requester is unprivileged and attempted to perform an operation that only a privileged requester may perform. For example, it may have attempted to act upon a consent record that it does not own, or it may have attempted to delete a consent record.

When using basic authentication, the requester must be listed in the Consent Service configuration **`service-account-dn`** property to be considered privileged.

Subject and actor do not match

Only a privileged requester can create or modify a consent record whose **`subject`** and **`actor`** values do not match.

Unindexed search

The Consent Service will not allow a client to make an unindexed search. In most cases, a client should be able to fix this by refining the search. For example, if a search by **`subject`** would be unindexed, perform a search by **`subject definition`** ID.

Search size limit exceeded

The Consent Service caps the maximum number of records that can be returned in a search result using its **`search-size-limit`** configuration property. This limit can be increased, or the client may be able to refine the search to produce fewer results.

Delegated Admin Application Guide

PingDirectory™ Product Documentation

Support

https://support.pingidentity.com/

Delegated Admin overview

Delegated Admin from Ping Identity Corp is an add-on to PingDirectory that enables the delegation of user and group management.

Introduction

Many organizations spend a disproportionate, and often wasteful, amount of time resetting passwords, updating account data, and completing other simple but recurring tasks. Delegated Admin lets organizations assign these responsibilities, as well as others associated with the management of identities in PingDirectory Server, to a subset of administrators. Such *delegated administrators* can be any user outside the organization's IT department, including a customer.

The following employees typically fulfill roles that involve at least a basic level of identity management:

- Help desk or customer service representatives who unlock and reset passwords
- Managers and Human Resources administrators who update employee profiles
- Application administrators who update identity attributes and manage access to applications

These employees represent strong candidates for inclusion in a group of delegated administrators.

Features

Delegated Admin lets delegated administrators complete the following tasks across groups, subtrees, and entire organizations:

- Create, view, and search user profiles
- View user account information, including account status, last login time, and password expiration date
- Update user attributes
- Implement constructed attributes

- Set attributes to `read-only`
- Enable and disable accounts
- Reset locked accounts
- Create and edit groups
- Manage the membership of groups and subgroups
- Manage the roles of users and groups
- Delete users, groups, and generic resource types
- Implement custom UI form fields
- Select user entries based on their Distinguished Names (DNs) without displaying the actual values of the DNs
- Preview and download reports about user profiles. Reporting provides the following features:
  - Capability to report for resources of a given type, or limited to members of a group
  - Ability to display multiple values per attribute for each user
  - Protection against spreadsheet formula injection
- Upload CSV files to add user, group membership, or ou records

Installing Delegated Admin

Depending on your environment and the location of your installation, different options and procedures are available when you install Delegated Admin. Before attempting to install the application, make certain that you complete the tasks in *Prerequisites* on page 1238, and in *Before you install* on page 1239.

Installation locations

Delegated Admin can be installed in any of the following locations:

- PingDirectory Server, including replicated instances
- PingDirectoryProxy Server
- External web server

The location that you choose determines the steps that you must perform to install Delegated Admin.

Prerequisites

Regardless of the location of your Delegated Admin installation, make certain the following Ping Identity products are installed and configured before attempting to install Delegated Admin.

Product	Description	Version
PingDirectory Server	Stores user-identity data. The HTTPS port that was configured during PingDirectory Server setup is required to install Delegated Admin. For information about upgrading PingDirectory Server, see *Upgrade PingDirectory Server* on page 1244. For information about installing and configuring PingDirectory Server, refer to *PingDirectory Server Administration Guide*.	See *Compatibility matrix* on page 1272.
PingFederate Server	Provides identities for authentication and authorization. For information about installing and configuring PingFederate Server, see *Configure PingFederate Server* on page 1273, or refer to *PingFederate Server Guide*.	See *Compatibility matrix* on page 1272.

Supported browsers

- Chrome
- Firefox
- Internet Explorer 11 and later

Obtaining the installation files

About this task

To obtain the Delegated Admin installation files, perform the following steps:

Steps

1. Download the Delegated Admin installation package, `pingdirectory-delegator-{version}.zip`, to the server on which you plan to install the application.
2. Extract the contents of the installation package.
3. Copy the folder named `/delegator` and its contents to the appropriate directory, as shown by the following table.

Server	Directory
PingDirectory Server	`/webapps`
Replicated instance of PingDirectory Server	`/webapps`
PingDirectoryProxy Server	`/webapps`
External web server	Directory for web-based apps

4. Copy the template file to the `config/account-status-notification-email-templates` folder of each instance of PingDirectory Server

   By default, the email is sent to the address that is found within the user's LDAP mail attribute.

ⓘ **Note:** A mail value must be provided for each user. For more information, refer to `common-header-fields.vm` in the email templates folder.

Before you install

If you plan to install Delegated Admin on PingDirectory Server or a replicated instance of PingDirectory Server, complete the relevant tasks in this section before installing the application.

PingDirectory Server

If you are installing Delegated Admin on a PingDirectory Server installation that had "Install with sample data" chosen as the installation option, remove the relevant Access Control Information (ACI) from the PingDirectory Server base entry. For more information, refer to the LDIF file `delegator/remove-sample-directory-data-aci.ldif`.

If you are installing Delegated Admin on a PingDirectory Server installation that did not have "Install with sample data" chosen as the installation option, proceed to *Obtaining the installation files* on page 1239.

Replicated instance of PingDirectory Server

If you plan to install Delegated Admin on a replicated instance of PingDirectory Server, perform the following steps:

1. Make certain that replication is enabled for the PingDirectory Servers.

   For information about enabling server replication, refer to *PingDirectory Server Administration Guide*.

2. To ensure that configuration changes are applied to all the servers in your topology, configure the PingDirectory Servers to use a configuration group called **`all-servers`**, as follows:

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

3. If you are installing Delegated Admin on a replicated PingDirectory Server that had "Install with sample data" chosen as the installation option, remove the relevant Access Control Information (ACI) from the PingDirectory Server base entry.

   Because replicated instances share the same data, perform this step against only one of the servers. For more information, refer to the LDIF file `delegator/remove-sample-directory-data-aci.ldif`.

   If you are installing Delegated Admin on a PingDirectory Server installation that did not have "Install with sample data" chosen as the installation option, proceed to *Obtaining the installation files* on page 1239.

Installing the application

About this task

The steps for installing Delegated Admin depend on whether you are setting up the application in a Unix/Linux environment or in a Windows environment.

> ⓘ **Note:** Regardless of your setup environment, if port 443 is used but not specified in the PingFederate Base URL, do not assign a value to `window.PF_PORT`.

Unix or Linux

About this task

To begin installing Delegated Admin in a Unix or Linux environment, run the following script in the `/delegator` directory from *Obtaining the installation files* on page 1239:

```
$ ./set-up-delegator.sh
```

The system generates a configuration file named `config.js` and a batch file named `delegated-admin.dsconfig`.

Windows

About this task

To begin installing Delegated Admin in a Windows environment, perform the following steps:

Steps

1. In the Delegated Admin application directory, copy or rename the file `example.config.js` to `config.js`.

   `config.js` contains comments and placeholders for necessary information. For example, the client ID that is required in this file must be one of the client IDs that has been defined for the PingFederate configuration. This value represents the client intended for token issuance, such as `dadmin`.
2. Open `config.js` in a text editor.

3. Change the variable values to match your setup configuration, as the following table shows.

`config.js` Variable	Value
`window.PF_HOST`	Public address of the PingFederate Server to which the application redirects the user's browser when logging on.
`window.PF_PORT`	PingFederate port number. If port 443 is used but not specified in the PingFederate Base URL, do not assign a value to `window.PF_PORT`.
`window.DADMIN_CLIENT_ID`	PingFederate Client ID for the application.

4. Save your changes to `config.js`.
5. Copy or rename the batch file `delegated-admin-template.dsconfig` to `delegated-admin.dsconfig`.
6. Open `delegated-admin.dsconfig` in a text editor and replace the variables (**`${variable}`**) with actual values.
7. Save your changes to `delegated-admin.dsconfig`.

All environments

Regardless of whether you are installing Delegated Admin in a Unix/Linux or Windows environment, perform the relevant steps in this section after you complete the previous OS-specific tasks.

PingDirectoryProxy Server

About this task

If you are installing Delegated Admin on PingDirectoryProxy Server, you must configure the Proxy instance using the **`delegated-admin.dsconfig`** script as described in *All locations except replicated PingDirectory Server instances* on page 1243.

In addition, you must perform the following steps to configure all instances of PingDirectory Server.

Steps

1. Modify `delegated-admin.dsconfig` by commenting out the following section:

```
*"Create an email account status notification handler for user creation."
```

   This modified file is run on PingDirectoryProxy Server.
2. Copy the batch file `delegated-admin.dsconfig` and name it something similar to `copy-of-delegated-admin.dsconfig` and open the copy in a text editor.
3. Remove the following elements and sections from the file:
   - Web-application-extension Delegator
   - Access-token-validator PingFederateValidator
   - Definition rest-resource-type
   - Definition delegated-admin-rights

4. Leave the following configuration elements in the copy exactly as they are configured on PingDirectoryProxy Server. These should be the only elements remaining in the copy:
   - Create an email account status notification handler for user creation (create-request-criteria and create-account-status-notification-handler).
   - Virtual-attribute Delegated Admin Privilege (set-virtual-attribute-prop)
   - Global ACI Authenticated access to the multi-update extended request for the Delegated Admin API (set-access-control-handler-prop)
   - Global ACI Authenticated access to the no-op request control for the Delegated Admin API (set-access-control-handler-prop)
5. Save your changes to the copy.
6. For each PingDirectory Server instance, copy the `PingDirectoryProxy/webapps/delegator/delegated-admin-account-created.template` template to the `PingDirectory/config/account-status-notification-email-templates/` directory.
7. Apply the commands from the copy of the batch file to all instances of PingDirectory Server as follows (for this example the copy is assumed to be named `copy-of-delegated-admin.dsconfig`):

```
$ ./bin/dsconfig \
--bindDN "cn=Directory Manager" \
--no-prompt \
--batch-file webapps/delegator/copy-of-delegated-admin.dsconfig \
--applyChangeTo server-group
```

Replicated instances of PingDirectory Server

About this task

If you are installing Delegated Admin on one or more replicated instances of PingDirectory Server, apply the following commands in `delegated-admin.dsconfig` to each instance:

```
$ ./bin/dsconfig \
  --bindDN "cn=Directory Manager" \
  --no-prompt \
  --batch-file webapps/delegator/delegated-admin.dsconfig  \
  --applyChangeTo server-group
```

External web server

About this task

If you are installing Delegated Admin on an external web server, perform the following steps:

Steps

1. Open `config.js` in a text editor.
2. Change the variable values to specify the location of PingDirectory Server, as the following table shows.

`config.js` Variable	**Value**
`window.DS_HOST`	host name of PingDirectory Server
`window.DS_PORT`	HTTPS port of PingDirectory Server

   To view an example outline that features these settings, refer to **`example.config.js`**.
3. Save your changes to `config.js`.
4. Open `delegated-admin.dsconfig` in a text editor.

5. Comment out the following lines, which are located in the `Configure the delegator web app` section, near the bottom of the file:

```
dsconfig create-web-application-extension --extension-name Delegator --
set base-context-path:/delegator --set document-root-directory:webapps/
delegator/app
dsconfig set-connection-handler-prop --handler-name "HTTPS Connection
 Handler" --add web-application-extension:Delegator
```

6. Save your changes to `delegated-admin.dsconfig`.
7. Create a CORS policy for the Delegated Admin HTTP servlet extension, where `<origin>` represents the public name of the host, proxy, or load balancer that presents the Delegated Admin web application:

```
dsconfig create-http-servlet-cross-origin-policy --policy-name "Delegated
 Admin Cross-Origin Policy" --set "cors-allowed-methods: GET" --set "cors-
allowed-methods: OPTIONS" --set "cors-allowed-methods: POST" --set "cors-
allowed-methods: DELETE" --set "cors-allowed-methods: PATCH" --set "cors-
allowed-origins: <origin>"
dsconfig set-http-servlet-extension-prop --extension-name "Delegated
 Admin" --set "cross-origin-policy:Delegated Admin Cross-Origin Policy"
```

All locations except replicated PingDirectory Server instances

About this task

ⓘ **Note:** The *PingDirectory*`/webapps/delegator/delegated-admin-account-created.template` template must be copied to the *PingDirectory*`/config/account-status-notification-email-templates/` directory. This file must be copied before running **`delegated-admin.dsconfig`** even if email functionality is not needed, since some commands in dsconfig require this template. See *Edit and copy the email template to PingDirectory Server* for information on setting up email.

To continue installing Delegated Admin on PingDirectory Server, PingDirectoryProxy Server, or an external web server, apply the following commands in `delegated-admin.dsconfig` to the appropriate server:

```
$ ./bin/dsconfig \
  --bindDN "cn=Directory Manager" \
  --no-prompt \
  --batch-file webapps/delegator/delegated-admin.dsconfig
```

Next steps

After you finish installing Delegated Admin, visit `https://webserverHost:httpPort/delegator` to view the application's **Sign On** page. At this time, you cannot log on to Delegated Admin because the rights of the delegated administrators have not been configured. For more information about configuring administrative rights, the REST resource type, session timeout values, and other properties, see *Configuring Delegated Admin* on page 1248.

After you configure Delegated Admin, *verify that the application is installed* and working successfully.

Upgrading Delegated Admin

Ping Identity periodically issues software with new features, enhancements, and fixes for improved performance. Administrators can use the PingDirectory Server's **`update`** tool to upgrade the current code version.

This chapter presents some update scenarios and the related implications to consider when upgrading your version of Delegated Admin.

Upgrade considerations

If you are running Delegated Admin version 3.5 or earlier, upgrade it to the latest version to use PingDirectory 8.0 or later.

For information about the compatibility between Delegated Admin and PingDirectory Server versions, see *Compatibility matrix* on page 1272.

Upgrade PingDirectory Server

Before attempting to upgrade Delegated Admin to the current version, ensure that you are running version 8.0.0.0 or later of PingDirectory Server. If you are already running PingDirectory Server 8.0.0.0 or later, proceed to *Upgrade the application* on page 1247.

For more information about the compatibility of Delegated Admin with different versions of PingDirectory Server and PingFederate Server, see *Compatibility matrix* on page 1272.

Overview and considerations

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

Consider the following when upgrading replicating servers:

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

> ⓘ **Tip:**
>
> For additional considerations, see the *Planning your upgrade guide*.

Upgrading servers in a topology

An update to the current PingDirectory Server release includes the introduction of a topology registry, which stores information that was stored previously in the admin backend, such as server instances, instance and secret keys, server groups, and administrator user accounts.

Before you begin

Before you can update and migrate the admin backend, the following conditions must be satisfied:

- Run the update tool and provide LDAP authentication options to the peer servers of the server being updated. The **`update`** tool connects to the peer servers of the server being updated to obtain the necessary information to populate the topology registry.

- On every server in the topology, configure the encryption protocol requested, either plain, TLS, StartTLS, or SASL, for an LDAP connection.
- Ensure the LDAP credentials are present on every server in the topology.
- Ensure the users related to the LDAP credentials have permissions to read from the admin backend and the config backend of every server in the topology. For example, you can use a root DN user that has `inherit-default-privileges` set to true, such as the `cn=Directory Manager` user, that exists on every server.
- The instance name is set on every server and is unique across all servers in the topology. The instance name is a server's identifier in the topology. After all servers in the topology have been updated, each server is uniquely identified by its instance name. After the name is set, it cannot be changed.

> ⓘ **Tip:**
>
> If needed, use the following command to set the instance name of a server prior to the update.
>
> ```
> $ bin/dsconfig set-global-configuration-prop \
>   --set instance-name:uniqueName
> ```

> ⓘ **Important:**
>
> This information is only applicable when updating from a server version that uses the admin backend to a server version that uses the topology registry. This is only the case when updating from a server version earlier than a 7.x to a 7.x version and is not applicable to any updates to a server version of 8.x or later.

About this task
Upgrade to a server that uses a topology registry to store network configuration, administrator user accounts, and keys information, as well as migrate the information from the previous server version.

Steps

1. To make clustered configuration changes in a mixed-version cluster, choose one of the following options:
   - Update each server to the same version.
   - Temporarily split up the cluster by changing the `cluster-name` property on the server instance configuration objects.

   > ⓘ **Note:**
   >
   > Changes to clustered configurations are not allowed in mixed-version clusters. This applies to configuration in the `cn=Cluster,cn=config` subtree and only applies to servers with matching cluster names.
2. Make clustered configuration changes again to mirror these changes across the topology.
3. Run the **`dsframework`** command on the server being updated.

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

```
  --set startTLSEnabled:true
```

> ⓘ **Important:**
>
> To run this command, you must have enabled or fixed the configuration of the LDAP Connection Handlers to support the desired connection security protocol on each server, so that its admin backend has the most up-to-date information.

Before allowing the update, the **`update`** tool verifies that the following conditions are satisfied on every server in the topology:

- When the first server is being updated, all other servers in the topology are online.
- When updating additional servers, all topology information was obtained from one of the servers that has already been updated.
- The users related to the provided LDAP credentials have read permissions to the config and admin backends of the peer servers.
- The instance name is set on every server and is unique across all servers in the topology.

> ⓘ **Note:**
>
> If any of these conditions or those listed in the preceding Before you begin section are not satisfied, the **`update`** tool lists all of the errors encountered for each server and provides instructions on how to fix them.

When all of these conditions are met, the cluster-wide configuration is synchronized on all servers in the topology.

> ⓘ **Note:**
>
> Older versions have some topology configuration under the **`cn=cluster,cn=config`** JSON attribute and field constraints. These items do not support mirrored cluster-wide configuration data. In an update, avoid custom configuration changes on a server being overwritten with the configuration on the mirrored subtree master. If additional configuration steps are needed, see step 4.

4. To synchronize the cluster-wide configuration data across all servers in the topology, run the **`config-diff`** tool on each pair of servers to determine the differences, and use **`dsconfig`** to update each instance using the **`config-diff`** output.

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

Upgrading PingDirectory Server

About this task

To upgrade PingDirectory Server, perform the following steps:

Steps

1. Download and extract the new version of PingDirectory Server to a location outside the existing server's installation.

   For these steps, assume that the existing server installation resides in `/prod/PingDirectory` , and that the new server version is extracted to `/home/stage/PingDirectory` .
2. To upgrade the existing PingDirectory Server, run the **`update`** tool that is provided with the new server package, as follows:

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

   If the tool detects any degree of customization, it might prompt for confirmation on server configuration changes.

Upgrade the application

In DA 3.3.0 and earlier, the setup script assigned a cross-origin resource sharing (CORS) policy to the Delegated Admin HTTP servlet extension. This policy is potentially insecure because the CORS setting **`Allowed-Origin`** permits requests that use a wildcard to allow requests from any origin. Unless you have made changes to secure this policy, remove it, as follows:

```
dsconfig set-http-servlet-extension-prop --extension-name "Delegated Admin"
 --reset "cross-origin-policy"
dsconfig delete-http-servlet-cross-origin-policy --policy-name "Delegated
 Admin Cross-Origin Policy"
```

ⓘ **Important:**

Beginning with Delegated Admin 3.2.0 and PingDirectory Server 7.2.1.0, the following configuration changes were made:

- **`delegated-admin-resource-type`** was replaced with **`rest-resource-type`**.
- **`delegated-administrator`** was replaced with **`delegated-admin-rights`** and **`delegated-admin-resource-rights`**.

As a result, Delegated Admin 3.0.2 or earlier requires PingDirectory Server 7.2.0.1 or earlier. Similarly, Delegated Admin 3.2.0 or later requires PingDirectory Server 7.2.1.0 or later.

The **`update`** tool converts earlier configurations to new configuration definitions. This tool is also used during the process of upgrading PingDirectory Server.

The migrated Delegated Admin configuration features a `group` REST resource type for the structural object classes `groupOfNames` and `groupOfUniqueNames`. If the original user's resource type configuration includes a value for `Org Search Filter`, then the migrated configuration also features a generic `orgs` REST resource type, with the structural object class `organizationalUnit` as the parent resource type of users. If necessary, change the structural object class on the resource type configuration after the Delegated Admin update completes.

ⓘ **Note:**

If you change the structural object class, you must stop the server to proceed with the update.

ⓘ **Note:**

The `delegated-admin-template.dsconfig` file has been updated to allow for `generate-password` extended requests and password validation details request controls. This change is not applied during an

update. You must run the following two **dsconfig** commands when updating PingDirectory Delegated Admin to Version 4.0.0:

```
dsconfig set-access-control-handler-prop --add \
'global-aci:(extop="1.3.6.1.4.1.30221.2.6.62")(version 3.0; \
acl "Authenticated access to the generate-password extended \
request for the Delegated Admin API"; allow (read) userdn="ldap:///all";)'
```

```
dsconfig set-access-control-handler-prop \
--add 'global-aci:(targetcontrol="1.3.6.1.4.1.30221.2.5.40")\
(version 3.0;acl "Authenticated access to the password validation details
 request \
control for the Delegated Admin API"; allow (read) userdn="ldap:///all";)'
```

Tip:

For additional considerations, see the *Planning your upgrade guide*.

To upgrade Delegated Admin on PingDirectory Server, perform the following steps:

1. Extract the contents of the Delegated Admin upgrade `.zip` file.
2. Rename the original **delegator** folder to retain a backup copy of the earlier version.
3. Copy the extracted folder named **delegator** to the PingDirectory Server folder named **webapps**.
4. Copy the **{OriginalDelegatorFolder}/app/config.js** configuration file to the new **delegator** folder.
5. Restart PingDirectory Server.

For more information, see the *PingDirectory Server Administration Guide*.

Configuring Delegated Admin

This chapter describes the necessary configuration to support Delegated Admin after the application is installed successfully.

At a minimum, you must configure the following properties on PingDirectory Server:

- Delegated administrator rights
- REST resource type
- Attributes and attribute searching

Configuration overview

Delegated Admin must have a PingDirectory Server and PingFederate Server installed. For installation instructions, refer to the documentation for each product.

The process of configuring support for Delegated Admin on PingDirectory Server includes the following tasks:

- Configure users as Delegated Admin administrators.
- Configure attributes and attribute searching.
- Configure groups whose management requires delegation

The process of configuring PingFederate Server includes the following tasks:

- Configure PingFederate as the identity provider for Delegated Admin.
- Configure PingFederate as the OAuth server for Delegated Admin.
- Register Delegated Admin as a client.

- Register PingDirectory Server as an OAuth token validator client.

Authentication configuration

The delegated administrator logs on to Delegated Admin through the PingFederate Server, which is configured as the authentication server and OpenID Connect (OIDC) provider. PingFederate validates the user's credentials against PingDirectory Server, encapsulates information **`claims`** about the user's identity, and issues an access token to Delegated Admin, which presents the token to PingDirectory Server in the HTTP Authorization request header.

Interaction with PingDirectory Server

PingDirectory Server is configured to accept access tokens by using Access Token Validators. The values that PingFederate Server sets for the access token **`sub`** claim must be mappable to a distinguished name (DN) in PingDirectory Server. Setting up an access token validator for use with Delegated Admin requires some coordination with the server configuration. In the suggested default configuration, the access token contains the entryUUID of the administrator user entry in the **`sub`** claim. This value is mapped back to a PingDirectory Server entry by using an Exact Match Identity Mapper.

Authorization by PingDirectory Server

After validation, PingDirectory Server checks the Delegated Admin configuration for authorization of the delegated administrator. Users or groups of users are authorized as delegated administrators in the PingDirectory Server Administrator Console, or with the **`dsconfig`** tool.

Configure authentication

One of the prerequisites to installing Delegated Admin is to configure the following OAuth clients within PingFederate:

- Delegated Admin, which obtains an OIDC token that describes the authenticated user

  For more information, see *Configure Delegated Admin as a new client (create OAuth client for Delegated Admin)* on page 1278.
- PingDirectory Server itself, which calls PingFederate to validate the OIDC token that Delegated Admin passes to it

  For more information, see *Configure PingDirectory Server as the token validator (create OAuth client for PingDirectory)* on page 1277.

Configuring delegated administrator rights on PingDirectory Server

About this task

To use Delegated Admin, an administrator must possess more than valid credentials and an access token that PingDirectory Server can validate. He or she must possess rights that are designated through the PingDirectory Server configuration. To delegate users or groups as administrators, use the PingDirectory Server Administrator Console (Delegated Admin rights and resource rights) or the **`dsconfig create-delegated-admin-rights`** and **`create-delegated-admin-resource-rights`** commands.

Admin Permissions

create

The administrator can create new resources of this type.

read

The administrator can read resources of this type.

Admin Permissions

> ⓘ **Note:**
>
> The create, delete, manage-group-membership, and update permissions require the read permission.

update

The administrator can edit resources of this type.

delete

The administrator can delete resources of this type.

reference

The administrator can reference resources when selecting a parent during the creation of another resource. With the reference permission specified, the administrator can use a parent REST resource type without seeing the option to manage the parent resource type. For example, if the parent type for users is Organizational Unit, the administrator can have reference rights to the Organizational Unit resource type only. The administrator can create users without seeing the **Manage Organizational Unit** navigation option.

The administrator can reference resource types in Delegated Admin attributes. For example, the administrator can select user entries from a list based on their DNs without displaying the actual values of the DNs.

manage-group-membership

The administrator can manage the membership of a group resource, by adding or removing members. This permission is only applicable to group resource types.

download

The administrator can download reports for resources of this type. With this permission, the **Download Report** button appears on the **Reporting** page for the administrator.

upload

The administrator can upload a CSV file to import resources of this type. With this permission, the **Upload File** button appears on the **Reporting** page for the administrator.

> ⓘ **Note:**
>
> For the parent resource type to be available for the creation of new entries under the parent, the read or reference permission must be specified.

To prevent changes that might break the configuration of the app, the app does not allow changes to RDN attributes of a resource entry DN, for resources referenced in the Delegated Admin server configuration. This includes the following configuration elements:

- `admin-user-DN` and `admin-group-DN` of Admin Rights
- `resource-subtree` and `resources-in-group` of Admin Resource Rights

For example, if an Admin Rights configuration contains `admin-group-DN: cn=Admin Group,dc=example,dc=com` and some administrator has rights to modify that particular group through the app, then the `cn` attribute of that group cannot be changed without invalidating the configuration. The attribute label will have a lock icon and a message indicating that the value can only be changed by a server administrator.

The example commands in this section illustrate the configuration options for delegated administration and are performed on PingDirectory Server.

ⓘ **Note:** Administrators who manage only specific subtrees cannot create users in an organization that does not reside under, or at the same level as, one of the subtrees.

Steps

- The following commands restrict an administrator to manage users in specified subtrees:

```
$ bin/dsconfig create-delegated-admin-rights \
  --rights-name admin1 \
  --set "admin-user-dn:uid=admin1,ou=people,dc=example,dc=com"
  --set enabled:true

$ bin/dsconfig create-delegated-admin-resource-rights \
  --rights-name admin1 \
  --rest-resource-type users \
  --set admin-scope:resources-in-specific-subtrees \
  --set "resource-subtree:ou=org1,dc=example,dc=com" \
  --set admin-permission:create \
  --set admin-permission:read \
  --set admin-permission:update \
  --set admin-permission:delete \
  --set enabled:true
```

- An administrator can be restricted to managing the member users of one or more specified groups. In the following example, we assume the existence of a static or dynamic group entry whose members include the users to be managed:

```
$ bin/dsconfig create-delegated-admin-rights \
  --rights-name admin1 \
  --set "admin-user-dn:uid=admin1,ou=people,dc=example,dc=com"
  --set enabled:true
$ bin/dsconfig create-delegated-admin-resource-rights \
  --rights-name admin1 \
  --rest-resource-type users \
  --set admin-scope:resources-in-specific-groups \
  --set "resources-in-group:cn=User Group,dc=example,dc=com" \
  --set admin-permission:read \
  --set admin-permission:update \
  --set enabled:true
```

- The Delegated admin must also have rights to a group REST resource type that matches the specified group. For more information, see *Manage groups*.
- Rather than delegate a single user as an administrator, you might find it more convenient to delegate an entire group of users as administrators, as follows:

```
$ bin/dsconfig create-delegated-admin-rights \
  --rights-name admin-group1 \
  --set "admin-group-dn:cn=Admin Group,ou=people,dc=example,dc=com"
  --set enabled:true

$ bin/dsconfig create-delegated-admin-resource-rights \
  --rights-name admin-group1 \
  --rest-resource-type users \
  --set admin-scope:all-resources-in-base \
  --set admin-permission:create \
  --set admin-permission:read \
  --set admin-permission:update \
  --set admin-permission:delete \
```

```
  --set enabled:true
```

In this example, groups can be configured to manage specific subtrees or groups with the **`resources-in-specific-subtrees`** or **`resources-in-group`** setting for the **`admin-scope`**. For more information about PingDirectory Server administrators and configuring dynamic and static groups, refer to the *PingDirectory Server Administration Guide*.

Parameterized Delegated Administrator Rights

Delegated Admin rights can be parameterized such that a single definition provides a pattern for new administrators. In this way a privileged administrator for a hosting company can use Delegated Admin to onboard a new tenant administrator to manage resources for the tenant's own organization. Using parameterized rights eliminates the need for Directory Server configuration changes to create a new administrator.

In the following example, it is assumed that there are three REST resource types configured: orgs, groups and users. The users resource type has the parent resource type orgs.

```
$ bin/dsconfig create-delegated-admin-rights \
--rights-name "Tenant Admin" \
--set enabled:true \
--set 'admin-group-dn:cn=($1),ou=groups,dc=example,dc=com'

$ bin/dsconfig create-delegated-admin-resource-rights \
--rights-name "Tenant Admin" \
--rest-resource-type users --set enabled:true \
--set admin-permission:create \
--set admin-permission:read --set admin-permission:update \
--set 'resource-subtree:ou=($1),dc=example,dc=com'

$ bin/dsconfig create-delegated-admin-resource-rights \
--rights-name "Tenant Admin" \
--rest-resource-type orgs --set enabled:true \
--set admin-permission:reference \
--set 'resource-subtree:ou=($1),dc=example,dc=com'
```

A privileged admin can perform the following steps to onboard a new tenant in Delegated Admin:

- Add a new org for the tenant.
- Add a new group with the same name as the new org, representing the tenant admins.
- Ads a new user representing an initial tenant admin.
- Add the new tenant admin user to the tenant admin group

The tenant admin user can now log in to the app and manage users for their own organization.

Configuring user self-service

About this task

PingFederate Server provides end users with the ability to self-service their own profiles. Additional configuration steps must be taken in both PingDirectory Server and PingFederate to enable users whom delegated administrators create to manage their own profiles through the PingFederate local identity profile-management feature.

ⓘ **Note:** Import the Ping Federate ldif, first in PingDirectoryProxy Server and then in PingDirectory Server. Constructed attributes need to be created only in PingDirectoryProxy Server. Creating and rebuilding indexes (part of the self-service configuration) is done on PingDirectory Server.

ⓘ **Note:**

Before running the **`dsconfig`** install file, copy the email template from the `delegator` directory to the PingDirectory server to its `config/account-status-notification-email-templates` directory. The system uses this template to send emails with the self-service link to the users.

Steps

1. Configure PingFederate for profile management.

   To allow users to change their passwords, enable Allow Password Changes in the HTML Form Adapter. You must make this change if you want to create passwords that the user must change on the first use.

   For example PingFederate configuration steps, see *Customer IAM configuration*.

   Setting up PingDirectory for customer identities in the PingFederate Administrator's Manual includes some of the following required steps on PingDirectory Server.
2. To create passwords that the user must change on the first use after account creation or a password reset, configure a PingDirectory password policy to force users to change their passwords.

   ```
   dsconfig set-password-policy-prop --policy-name "Default Password Policy"
    \
   --set force-change-on-add:true --set force-change-on-reset:true
   ```

   This policy requires that you enable Allow Password Changes as mentioned above.

   With these changes, when a user signs on to the PingFederate self-service page, the page prompts the user to change their password.
3. Import the required additional LDAP schema provided by PingFederate into PingDirectory Server.
   a. On PingFederate Server, copy the LDIF file `local-identity-pingdirectory.ldif` from the following location: *<pf_install>*`/pingfederate/server/default/conf/local-identity/ldif-scripts/local-identity-pingdirectory.ldif`.
   b. Use the `scopy` command to securely copy the LDIF file to your local machine.
4. Update the LDAP schema.
   a. Log on to the PingDirectory Server Administrator Console.
   b. Go to **LDAP Schema**# **Schema Utilities**.
   c. Click **Import Schema Element**.
   d. Copy the schema changes from the file *<pf_install>*`/pingfederate/server/default/conf/local-identity/ldif-scripts/local-identity-pingdirectory.ldif`.
   e. Paste the schema changes into the text area.
   f. Click **Import**.
5. Create an equality index for the pf-connected-identity attribute.

   ```
   $ bin/dsconfig create-local-db-index \
     --backend-name userRoot \
     --index-name pf-connected-identity \
     --set index-type:equality
   ```
6. After adding the index, use the rebuild-index utility to build the indexes. For instance, the following sample builds the required index.

   ```
   $ bin/rebuild-index \
     --baseDN "dc=example,dc=com" \
     --index pf-connected-identity
   ```

7. Configure PingDirectory Server Composed Attributes.

   In previous versions of Delegated Admin, the remaining configuration was achieved by setting a constructed attribute on the user REST resource type. In the latest version, composed attribute plugins should be used instead as they provide the following advantages:

   - The populate-composed-attribute-values tool can be used to enable self-service for any existing users.
   - Self-service will be enabled for any users not created through the Delegated Admin app.

   Configure two Composed Attribute Plugins as follows:

   > ⓘ **Note:**
   >
   > *<users-base-dn>* and *<users-object-class>* must be replaced with the search base DN and structural object class of your REST Resource Type.

   ```
   $ bin/dsconfig create-plugin \
     --plugin-name pf-connected-identities \
     --type composed-attribute \
     --set enabled:true \
     --set attribute-type:objectClass \
     --set value-pattern:pf-connected-identities \
     --set target-attribute-exists-during-initial-population-behavior:merge-
   existing-and-composed-values \
     --set "include-base-dn:<users-base-dn>" \
     --set "include-filter:(objectClass=<users-object-class>)"

   $ bin/dsconfig create-plugin \
     --plugin-name pf-connected-identity \
     --type composed-attribute \
     --set enabled:true \
     --set attribute-type:pf-connected-identity \
     --set "value-pattern:auth-source=pf-local-identity:user-id={entryUUID}"
    \
     --set "include-base-dn:<users-base-dn>" \
     --set "include-filter:(objectClass=<users-object-class>)"
   ```

   If you configure composed attribute plugins as described after upgrading an existing deployment, then you should remove the old constructed attribute configuration as follows:

   ```
   $ bin/dsconfig set-rest-resource-type-prop --type-name users \
     --remove auxiliary-ldap-objectclass:pf-connected-identities \
     --remove post-create-constructed-attribute:pf-connected-identity \
     --remove update-constructed-attribute:pf-connected-identity
   ```

8. Optional: To enable self-service for any existing users not already linked to PingFederate:

   ```
   $ bin/populate-composed-attribute-values -h <host> -p <port> -D
    "cn=Directory Manager" -w <password>
   ```

Configuring attributes and attribute search on PingDirectory Server

About this task

The file that installs Delegated Admin also specifies the following values:

- Object class of user entries through **`structural-ldap-objectclass:inetOrgPerson`**

- Number of user attributes to expose

> ⓘ **Note:** Delegated Admin supports the following attribute types:
>
> - `Boolean`
> - `Integer`
> - `String`
> - `DateTime`
> - Distinguished Name (DN)
> - Custom attributes
> - Constructed attributes
> - Multivalued attributes

Steps

1. If necessary, change the attribute that is designated as the primary attribute.

```
$ bin/dsconfig set-rest-resource-type-prop \
  --type-name users \
  --set primary-display-attribute-type:mail
```

2. Configure any additional user attributes to appear in Delegated Admin by specifying the LDAP attribute type to expose and by providing a display name for it.

```
$ bin/dsconfig create-delegated-admin-attribute \
  --type-name users \
  --attribute-type customAttr
  --set "display-name:My custom attribute"
```

3. Configure attributes with DN syntax on resource types to provide a reference from one resource to another.

   Such an attribute is the standard LDAP `manager` attribute.

   The referencing resource does not have to be the same type of resource as the referenced resource. Delegated Admin allows the referenced resource to be selected without displaying the actual value of the DN.

   In this example, the `manager` attribute is included in the users resource type, and its value is constrained to reference only resources of type `managers`. The `managers` REST Resource Type is assumed to have already been defined.

```
$ bin/dsconfig create-delegated-admin-attribute \
  --type-name users \
  --attribute-type manager \
  --set display-name:Manager \
  --set reference-resource-type:managers
```

   In addition, the Delegated Admin resource rights for the administrator must provide either read or reference permission to `managers`.

```
$ bin/dsconfig create-delegated-admin-resource-rights \
  --rights-name Admin \
  --rest-resource-type managers \
  --set enabled:true \
  --set admin-permission:reference \
  --set admin-scope:all-resources-in-base
```

   For more information about resource rights and permissions, see *Configuring delegated administrator rights on PingDirectory Server* on page 1249.

4. Use the following command to set the search filter, where `%%` represents the search text entered in the web application:

```
$ bin/dsconfig set-rest-resource-type-prop \
  --type-name users \
  --set 'search-filter-pattern:(|(cn=*%%*)(mail=%%*)(uid=%%*))'
```

   When search text is entered in Delegated Admin, the property **`search-filter-pattern`** specifies which attributes to search in PingDirectory Server. To satisfy the query, define the appropriate attribute indexes for PingDirectory Server. For more information, refer to *PingDirectory Server Administration Guide*.

5. To manage users whose profiles feature a large number of attributes, place the attributes in logical groupings, called *attribute categories*, and give them a specific display order.

   The following commands create attribute categories and specify their display order:

```
$ bin/dsconfig create-delegated-admin-attribute-category \
  --display-name "Basic Information" \
  --set display-order-index:1

$ bin/dsconfig create-delegated-admin-attribute-category \
  --display-name "Contact Information" \
  --set display-order-index:2

$ bin/dsconfig create-delegated-admin-attribute-category \
  --display-name "Other Attributes" \
  --set display-order-index:3
```

6. The following example commands assign attributes to a category and specify the display order of each attribute within its category.

```
$ bin/dsconfig set-delegated-admin-attribute-prop \
  --type-name users \
  --attribute-type cn \
  --set "attribute-category:Basic Information" \
  --set display-order-index:1

$ bin/dsconfig set-delegated-admin-attribute-prop \
  --type-name users \
  --attribute-type sn \
  --set "attribute-category:Basic Information" \
  --set display-order-index:2
```

   Unassigned attributes are displayed in a miscellaneous category.

7. For multivalued LDAP attributes, indicate whether the application should present them as multivalued. If not specified, the attributes are presented in the application as single-valued, even if the LDAP schema definition for the attribute allows multiple values. Note that this setting does not apply to attributes that are handled by custom UI form fields.

```
$ bin/dsconfig set-delegated-admin-attribute-prop \
 --type-name users \
 --attribute-type mail \
 --set multi-valued:true
```

Constructed attributes

A *constructed attribute* is an attribute whose value is computed from values that are assigned to other attributes. For example, the system might construct a full- or common-name attribute, `cn`, from values that are assigned to the standard `givenName` and `sn` attributes, as follows:

```
dsconfig create-constructed-attribute
  --attribute-name ReqConstructedCN --set attribute-type:cn \
```

```
  --set 'value-pattern:{givenName} {sn}'
```

Beginning with Delegated Admin 3.5.0 and PingDirectory Server 7.3.0.1, the value of a constructed attribute can be updated automatically whenever the value of a source attribute is not only created, but also when it is edited.

```
dsconfig set-rest-resource-type-prop \
  --type-name users  \
  --set post-create-constructed-attribute:ReqConstructedCN  \
  --set update-constructed-attribute:ReqConstructedCN
```

In these examples, a change to the value of `givenName` or `sn` forces a corresponding change to the value of `cn`. Attributes that contribute to a required constructed attribute are identified in the UI as **Required**, even if they were not originally designated as such. Because `cn` is a required attribute in this example, `givenName` and `sn` are also required.

ⓘ **Note:** The capability of an attribute to be changed after its creation is referred to as its *mutability*.

As with standard attributes, constructed attributes are stored as LDAP attributes in a database like PingDirectory Server.

Setting an attribute to read-only

About this task

Beginning with Delegated Admin 3.5.0 and PingDirectory Server 7.3.0.1, user access to standard and constructed attributes can be set to `read-only` as well as to `read/write`. We recommend that you restrict access to constructed attributes to `read-only`. Read-only attributes do not appear on the UI pages that are associated with the creation of users groups and other objects.

Use the **`dsconfig`** tool to set a standard or constructed attribute as `read-only`, as the following example shows:

```
dsconfig set-delegated-admin-attribute \
  --type-name users  \
  --attribute-type modifyTimestamp  \
  --set mutability:read-only
```

The following example resets a standard or constructed attribute from `read-only` to `read/write`:

```
dsconfig set-delegated-admin-attribute \
  --type-name users  \
  --attribute-type modifyTimestamp  \
  --reset mutability
```

Users and groups

Delegated administrators can be configured to manage users and groups in PingDirectory Server, as follows:

- Create new entries
- Read, view, and search existing entries
- Edit and update existing entries

The following sections describe users and groups in more detail.

Enable user creation

Enable the creation of new users and resources by configuring either a parent entry distinguished name (DN) or parent resource type where new users will be located. If you configure a parent DN, the entry that it references must exist in PingDirectory Server. All new users are created in this single location. If necessary, use **`ldapmodify`** to create the parent entry. For more information about the **`ldapmodify`** tool or about command-line help, refer to the *PingDirectory Server Administration Guide*. Alternatively, if a parent resource type is configured, the administrator can choose the specific resource where the new user is created.

ⓘ **Note:** Delegated Admin cannot list organizations in which the delegated administrator is unable to manage user entries. Further, administrators who manage only specific subtrees cannot create users in an organization that does not reside under, or at the same level as, one of the subtrees.

The following example specifies a single location for new users on PingDirectory Server:

```
$ bin/dsconfig set-rest-resource-type-prop \
  --type-name users \
  --set "parent-dn:ou=people,dc=example,dc=com" \
  --reset parent-resource-type
```

The setup script creates a resource type named **`orgs`**, which works with entries that feature the **`organization objectClass`**.

The following example shows that any organization resource can function as the location for new users on PingDirectory Server:

```
$ bin/dsconfig set-rest-resource-type-prop \
   --type-name users \
   --reset parent-dn \
   --set parent-resource-type:orgs
```

A different resource type can be created for **`organizationalUnit objectClass`** entries, as follows:

```
$ bin/dsconfig create-rest-resource-type \
  --type-name orgUnits \
  --set "display-name:Organizational Units" \
  --set primary-display-attribute-type:ou \
  --set "search-filter-pattern:(&(objectClass=organizationalUnit)(ou=%%))" \
  --set structural-ldap-objectclass:organizationalUnit \
  --set enabled:false

$ bin/dsconfig create-delegated-admin-attribute \
  --type-name orgUnits \
  --attribute-type ou \
  --set "display-name:Organizational Unit"

$ bin/dsconfig set-rest-resource-type-prop \
  --type-name orgUnits \
  --set enabled:true
```

The new resource type can be referenced as a **`parent-resource-type`**.

By default, new entries are named by their server-generated **`entryUUID`** values. To change this behavior, configure the LDAP **`RDN`** attribute.

ⓘ **Note:** The **`RDN`** attribute type must also be configured as a Delegated Admin attribute. For more information, see *Configuring attributes and attribute search on PingDirectory Server* on page 1254. Do not set read-only attributes as the **`RDN`** attribute.

In the following example, **uid** names new entries and becomes a required attribute:

```
$ bin/dsconfig set-rest-resource-type-prop \
  --type-name users \
  --set create-rdn-attribute-type:uid
```

New users are always created with their configured structural LDAP **objectclass**. One or more auxiliary LDAP **objectclasses** can be specified, as the following example shows:

```
$ bin/dsconfig set-rest-resource-type-prop \
  --type-name users \
  --set auxiliary-ldap-objectclass:ubidPersonAux
```

When existing users without all of the specified auxiliary **objectclasses** are edited, the missing **objectclasses** are updated automatically.

By default, the password field appears at the top of the **Create User** form outside any category. The next example adds the password field to the "Required fields" category and sets the location of the field to 2.

```
dsconfig set-rest-resource-type-prop \
    --type-name users \
    --set "password-attribute-category:Required fields" \
    --set password-display-order-index:2
```

ⓘ **Note:** If the server does not require a password, you can leave the password field blank when you create a user. Without a password, the user cannot sign on to *Configuring user self-service* on page 1252.

Enabling Account Information tab content

The Delegated Admin GUI's **Account Information** tab provides information about a user account. For Delegated Admin to display the user account information, you must enable the Password Policy State JSON virtual attribute for the users object class. You can then configure the information that appears.

Steps

1. For each PingDirectory instance that contains users, enable the Password Policy State JSON virtual attribute for the users object class.

   ⓘ **Note:** You do not need to enable this virtual attribute on PingDirectoryProxy instances.

   For example, the following command enables the virtual attribute for users with the person object class, which includes users whose REST resource type structural object class is derived from person (for example, inetOrgPerson).

   ```
   $ bin/dsconfig set-virtual-attribute-prop \
        --name "Password Policy State JSON" \
        --set enabled:true \
        --set require-explicit-request-by-name:true \
        --set "filter:(objectClass=person)" \
   ```

```
     --no-prompt --applyChangeTo server-group
```

After you enable the virtual attribute, delegated administrative users can access account information for a user in the Delegated Admin GUI.

The **Account Information** tab provides account status by default. To display the last login time and the password expiration date, you must set their properties. You configure these items per password policy.

When not configured, these entries appear as follows.

LAST LOGIN

Last login time not available

(However, that entry is also given when the user has not logged in.)

PASSWORD EXPIRATION

Password expiration date has not been enabled

You can configure these items in the Administrative Console or by using the **dsconfig** tool interactively or noninteractively. The steps below use the noninteractive approach.

2. Decide the password policy for which you want to enable the last login time and password expiration date.

   For more information, see *Managing Password Policies* on page 639.

   ```
   dsconfig list-password-policies
   ```

3. Decide whether to include the last login time.

   To include the last login time, decide which property to set. You can set either of the following properties.

   - `maximum-recent-login-history-successful-authentication-count`
   - `last-login-time-format`

     If you use this property, make sure the `last-login-time-attribute` has its default value `ds-pwp-last-login-time`.

     Values for `last-login-time-format` include:

     - `yyyyMMddHHmmss'Z'` for second-level accuracy
     - `yyyyMMdd` for day-level accuracy

4. Decide whether to include the password expiration date.

   To include this information, set the following property.

   `max-password-age`

5. Set the desired password policy properties.

   For example:

   ```
   $bin/dsconfig set-password-policy-prop \
        --policy-name "<password_policy_name>" \
        --set maximum-recent-login-history-successful-authentication-
   count:<count_value> \
        --set "max-password-age:<password_age_value>" \
        --no-prompt --applyChangeTo server-group
   ```

Manage groups

Use the following commands to delegate a user as a group administrator:

```
$ bin/dsconfig create-delegated-admin-rights \
```

```
  --rights-name group-admin1 \
  --set "admin-user-dn:uid=admin1,ou=people,dc=example,dc=com"
  --set enabled:true

$ bin/dsconfig create-delegated-admin-resource-rights \
  --rights-name group-admin1 \
  --rest-resource-type groups \
  --set admin-scope:resources-in-specific-subtrees \
  --set "resource-subtree:ou=Groups,dc=example,dc=com" \
  --set admin-permission:manage-group-membership \
  --set admin-permission:create \
  --set admin-permission:read \
  --set admin-permission:update \
  --set admin-permission:delete \
  --set enabled:true

$ bin/dsconfig create-delegated-admin-resource-rights \
  --rights-name group-admin1 \
  --rest-resource-type users \
  --set admin-scope:resources-in-specific-subtrees \
  --set "resource-subtree:ou=org1,dc=example,dc=com" \
  --set admin-permission:read \
  --set enabled:true
```

The administrative scope for users determines which users are visible to the group administrator. In this example, all users in the subtree **`ou=org1,dc=example,dc=com`** are visible. An administrator can be configured to edit users as well as to manage group memberships.

The group administrator can view, add, and remove any of the users within their administrative scope to the membership of groups within the groups' administrative scope. Static groups can be nested. Users who belong indirectly to a group through nesting are visible as group members but cannot be removed. Users can be removed only from the group of which they are a member. For example, an Employees group might include a Developers group as a nested member. In such a scenario, a user in the Developers group is a direct member of that group and an indirect member of Employees. This member can be removed only when viewing the Developers group, not when viewing the Employees group.

If a group is configured as a dynamic or virtual static group rather than a static group, then the group and its members are visible, but the group membership cannot be modified.

Set group attributes

The default settings for group attributes specify **`cn`** and **`description`** as group attributes, with **`cn`** used for the group title in Delegated Admin. To create the default settings, use the following commands with a search DN and parent DN (**`"dc=example,dc=com"`**):

```
$ bin/dsconfig create-rest-resource-type \
  --type group \
  --type-name groups \
  --set "display-name:Groups" \
  --set enabled:false \
  --set "search-base-dn:dc=example,dc=com" \
  --set primary-display-attribute-type:cn \
  --set resource-endpoint:groups \
  --set "search-filter-pattern:(cn=*%%*)" \
  --set structural-ldap-objectclass:groupOfUniqueNames
  --set parent-dn:dc=example,dc=com

$ bin/dsconfig create-delegated-admin-attribute \
  --type-name groups \
  --attribute-type cn \
  --set "display-name:Name"
```

```
$ bin/dsconfig create-delegated-admin-attribute \
  --type-name groups \
  --attribute-type description \
  --set "display-name:Description"

$ bin/dsconfig set-rest-resource-type-prop \
  --type-name groups \
  --set enabled:true
```

Set group search filter

When entering text to search for groups, the groups' **`search-filter-pattern`** property specifies the attributes to be searched in PingDirectory Server. To satisfy the query, define the appropriate attribute indexes for PingDirectory Server. The default setting searches the attribute **`cn`** for the search text, which is represented by **`%%`**. Use the following command to set the group search filter:

```
$ bin/dsconfig set-rest-resource-type-prop \
  --type-name groups \
  --set 'search-filter-pattern:(cn=*%%*)'
```

For more information about managing groups, refer to *PingDirectory Server Administrator Guide*.

Create a group

Users can be added as members to groups that delegated administrators create and manage. Subgroups can also be added as members to a group.

The configuration for each delegated group type consists of the following elements:

- Group REST resource type – Defines the attributes to locate groups in the directory information tree (DIT).
- Parent DN or Parent resource type – Specifies the location in which to create groups in the DIT.
  - To specify a Parent DN for a resource type, type the value in the **Parent DN** text box in the **Resource Creation** section. The Parent DN is often identical to the Search Base DN, such as `ou=customers,ou=Groups, dc=example,dc=com`.
  - To specify a Parent resource type, select a value from the **Parent Resource Type** list box in the **Resource Creation** section. Delegated administrators are subsequently presented with a list box that lets them select a resource, and the group is created under the selected parent resource. If you specify a Parent resource type, set a value for the **Primary Display Attribute Type** in the **Delegated Admin** section. This setting determines the values that are displayed in the Delegated Admin GUI. For example, a Primary Display attribute type of `ou` displays the `ou` value in the list box for each resource within the Parent resource type.
- Attributes to present to the delegated administrators.

To configure a Group REST resource type, use the **Edit Group REST Resource Type** page in the PingData Administrator Console. The **Search Base DN** value in the **General Configuration** section determines the data structure that is searched in Delegated Admin, and the **Display Name** value in the **Delegated Admin** section specifies the label of the REST resource in the Delegated Admin GUI.

PingData Administrator Console		Delegated Admin
UI Form Field	**Page and Section**	**UI Form Field on Create a New Group Page**
`Display Name`	**General Configuration**# **Delegated Admin**	`Select a Type` label
`REST Resource Type`	**Edit Delegated Admin Rights**# **Delegated Admin Resource Rights**	`Select a Type` option

PingData Administrator Console		Delegated Admin
UI Form Field	**Page and Section**	**UI Form Field on Create a New Group Page**
`Parent Resource Type`	**General Configuration**# **Resource Creation**	Display name for Parent resource type
`Display Name`	**General Configuration**# **Delegated Admin Attributes**	Additional elements like `CN`, `Description`, `Business Category`, and `Organization`

Adding a user to a group

Users can be added to groups from the **Manage Users** page as well as from the **Manage Groups** page. When the Delegated admin rights for a User REST resource type have admin scope resources-in-specific-groups, a user is added to one of the configured groups when the user is created.

Adding a new user to a configured group

About this task

When the Delegated admin rights for a User REST resource type have admin scope resources-in-specific-groups (and only in this case) a user is added to one of the configured groups when the user is created.

- For admins having rights to only one group, the new user is automatically added to that group. No field for **Select Group** will display.
- For admins having rights to more than one group, the admin selects a group to add the user to in the **Select Group** list.
- Admins can select from both Static and Dynamic groups.
- For Dynamic groups, in addition to selecting the group, the new entry must also match criteria for membership of that group. For example, a Dynamic group has members with uid=user.111*. The uid starts with **user.111**.

In order for the Admin to create a new user in that group they need to:

Steps

1. Select the group name from the list.
2. Enter the value for uid that starts with **user.111**.

Adding a user from the Manage Users page

About this task

To add a user from the **Manage Users** page, perform the following steps:

Steps

1. In Delegated Admin, click **Manage Users**.
2. Select or search for the user to add to a group.
3. Expand the user profile.
4. Click **Edit**.
5. Click **Groups**.
6. Select or search for the appropriate group.
7. From the **Nonmember Groups** list, click **+** to the right of the group.
   The group is moved to the **Member Groups** list.

Adding a user from the Manage Groups page

About this task

To add a user from the **Manage Groups** page, perform the following steps:

Steps

1. In Delegated Admin, click **Manage Groups**.
2. Select or search for the appropriate group.
3. Expand the group profile.
4. Click **Edit**.
5. Select or search for the appropriate user.
6. From the **Nonmembers** list, click **+** to the right of the user.
   The user is moved to the **Members** list.

Generic resource types

Use *generic resource types* to manage LDAP entries that are neither users nor groups, as follows:

- Create new entries
- Read, view, and search existing entries
- Edit and update existing entries

A generic resource type can represent a device, for example, or can be used for organizational unit branch entries that are parents of other resources.

Defining a generic resource type

About this task

Generic resources can be defined for any structural LDAP object class, and can function as members of a group. The following example enables the management of device entries:

```
$ bin/dsconfig create-rest-resource-type \
  --type-name device \
  --set enabled:true \
  --set resource-endpoint:device \
  --set "display-name:Device" \
  --set structural-ldap-objectclass:device \
  --set search-base-dn:dc=example,dc=com \
  --set parent-dn:dc=example,dc=com \
  --set 'search-filter-pattern:(cn=*%%*)' \
  --set primary-display-attribute-type:cn

$ bin/dsconfig create-delegated-admin-attribute \
  --type-name device \
  --attribute-type cn \
  --set "display-name:Device Name" \
  --set display-order-index:1

$ bin/dsconfig create-delegated-admin-attribute \
  --type-name device \
  --attribute-type serialNumber \
  --set "display-name:Serial Number" \
  --set display-order-index:2
```

After you define a generic resource type, create Delegated Admin resource rights for it. Generic resource administrators must have read access to the user resource. For more information, see *Configuring delegated administrator rights on PingDirectory Server* on page 1249.

Differentiating resource types within the same subtree

About this task

If you have resource types with the same object class and whose subtrees overlap, you can differentiate them through include filters. An entry is categorized as a given resource type only if the entry's attributes match the resource type's include filters.

You can create a new resource only if the resource attributes match the include filters for the type of resource being created.

For example, to differentiate between two kinds of organizationalUnit resources stored in the same subtree, run the following commands.

```
$ bin/dsconfig create-constructed-attribute \
  --attribute-name businessCategoryHotel \
  --set attribute-type:businessCategory \
  --set value-pattern:Hotel

$ bin/dsconfig create-constructed-attribute \
  --attribute-name businessCategoryCruiseLine \
  --set attribute-type:businessCategory \
  --set value-pattern:CruiseLine

$ bin/dsconfig create-rest-resource-type \
  --type-name Hotel \
  --set enabled:true \
  --set resource-endpoint:hotels \
  --set "display-name:Hotel" \
  --set structural-ldap-objectclass:organizationalUnit \
  --set search-base-dn:dc=example,dc=com \
  --set parent-dn:dc=example,dc=com \
  --set 'search-filter-pattern:(&(objectClass=organizationalUnit)(ou=%%))' \
  --set primary-display-attribute-type:ou \
  --set include-filter:(businessCategory=Hotel) \
  --set post-create-constructed-attribute:businessCategoryHotel

$ bin/dsconfig create-rest-resource-type \
  --type-name "Cruise Line" \
  --set enabled:true \
  --set resource-endpoint:cruiselines \
  --set "display-name:Cruise Line" \
  --set structural-ldap-objectclass:organizationalUnit \
  --set search-base-dn:dc=example,dc=com \
  --set parent-dn:dc=example,dc=com \
  --set 'search-filter-pattern:(&(objectClass=organizationalUnit)(ou=%%))' \
  --set primary-display-attribute-type:ou \
  --set include-filter:(businessCategory=CruiseLine) \
  --set post-create-constructed-attribute:businessCategoryCruiseLine
```

Configuring a resource's summary display in the Delegated Admin GUI

The Delegated Admin GUI provides a summary of attributes for a resource. For example, the profiles for the user and group resources show a summary. By default, a resource's summary shows all required

attributes and all search attributes for the resource. However, you can explicitly set the items to include in the summary.

About this task

If you configure any attributes to be in the summary, only those attributes appear in the summary.

You configure an attribute to appear in the summary by using **`dsconfig`** to set the `include-in-summary` property or by using the Administrative Console to select the **Include In Summary** check box.

Steps

- Given a resource, for each attribute to include in the summary for that resource, configure the attribute.
  - Using `dsconfig`

    Use the **`dsconfig set-delegated-admin-attribute-prop`** subcommand, as explained below.

    ```
    $ bin/dsconfig set-delegated-admin-attribute-prop \
      --type-name <name_of_type> \
      --attribute-type <type_of_attribute> \
      --set include-in-summary:true
    ```

    where

    <name_of_type> is the resource type.

    <type_of_attribute> is the name of an existing attribute for the given resource.

    For example:

    ```
    $ bin/dsconfig set-delegated-admin-attribute-prop \
      --type-name users \
      --attribute-type cn \
      --set include-in-summary:true
    ```

    To remove an attribute from the summary, use the same command but change `true` to `false`.
  - Using the PingDirectory Administrative Console

    1. Sign on to the PingDirectory Administrative Console.
    2. From the **Configuration** screen, click **REST Resource Types**.
    3. Edit a type.
    4. Go to the **Delegated Admin attributes** section.
    5. Edit the attribute and select its **Include In Summary** check box.

    To remove an attribute from the summary, clear its **Include In Summary** check box.

Customizing UI form fields

About this task

Delegated Admin provides a way to exercise additional control over the form field for a given attribute. Although the presentation can be customized completely, example files are provided to help give a sense of the capabilities from a functionality standpoint.

ⓘ **Note:** Because the example files will most likely require modifications to address your specific needs, we recommend a basic level of familiarity with HTML, CSS, and JavaScript when dealing with custom UI form fields.

The example files cover the following scenarios:

- Single-selection lists, which allow users to select one item from multiple options
- Check boxes, which can be customized so that one check box is dependent on the other
- String fields with validation and error messages
- Custom, user-friendly display of a JSON attribute, such as ubidEmailJSON
- Multivalue string field, which allows users to enter multiple, comma-separated values

To set up Delegated Admin to use the custom HTML files and to display custom UI form fields, perform the following steps:

Steps

1. Navigate to `PingDirectory/webapps/delegator/app/customAttributes`.
2. Use an editor to open the relevant example HTML file, as shown by the following table.

UI Form Field	Example File
Single selection list	`example-select.html`
Dependent check boxes	`example-dependent.html`
String field	`example-basic.html`
Custom display of a JSON attribute	`example-json.html`
Multivalue string field	`example-multivalue.html`

3. Make the appropriate modifications to the example file.

   For information about resizing and other custom capabilities, refer to the comments in `PingDirectory/webapps/delegator/app/assets/iframe--v1.js`.
4. Save the example file as a new file named `attributeName.html`.
   For example, if you are setting up a custom attribute presentation for `cn`, name the file `cn.html`.
5. *Optional:* For further clarity, create a directory under the `customAttributes` directory with the name of the resource endpoint of the REST resource types.
   For example, because a `users` type exists with the endpoint `usersEndpoint`, you might place `cn.html` under `customAttributes/usersEndpoint/`.

   When locating the appropriate HTML file, Delegated Admin searches for the attribute name under the endpoint folder, if one exists, before searching under the `customAttributes` folder.
6. Within PingDirectory Server, set the `attribute-presentation` field for the `Delegated Admin` attribute to `custom`, as the following example shows.

```
dsconfig set-delegated-admin-attribute-prop \
  --type-name users  \
  --attribute-type cn  \
  --set attribute-presentation:custom
```

Next steps

Browsers that attempt to aggressively cache HTML files often save and reuse outdated versions of those files. Because such browsers might not display changes that are made to an example HTML file, we recommend that you include the following `<meta>` tag in the head of each custom HTML file:

```
<meta http-equiv="Cache-Control" content="no-cache"/>
```

If your browser still does not display the updated HTML file, try one or more of the following suggestions:

- Clear the browser's cache.
- Use a private browsing window.
- Try a different browser.

Setting up email invitations for a new user

About this task

To set up email invitations for a new user, complete the following tasks:

1. Set up PingFederate for local identity profile management. For more information, see *Configure profile management by users* and *Configuring user self-service* on page 1252.

   When you complete this task, the PingFederate configuration will have a Local Identity Profile.
2. Configure Delegated Admin profile management by users. For more information, see *Configure profile management by users* and *Configuring user self-service* on page 1252.

   When you complete this task, users whom the Delegated Admin creates have the pf-connected-identities auxiliary object class as well as a pf-connected-identity attribute value, providing integration with PingFederate's user self-service.
3. Instruct users to copy the email template to PingDirectory Server. For more information, see *Editing and copying the email template to PingDirectory Server* on page 1268.
4. Create request criteria to match Delegated Admin user **ADD** requests. For more information, see *Creating request criteria to match Delegated Admin user ADD requests* on page 1269.
5. Edit the provided email template and insert the URL to the PingFederate self-service profile management endpoint. For more information, see *Editing and copying the email template to PingDirectory Server* on page 1268.
6. Create an SMTP external server. For more information, see *Creating an SMTP external server* on page 1269.
7. Create a multi-part Email Account Status notification handler for Delegated Admin user **ADD** requests. For more information, see *Creating a multi-part Email Account Status notification handler for Delegated Admin user ADD requests* on page 1269.

The following sections describe these tasks in more details.

Editing and copying the email template to PingDirectory Server

About this task

An example email template is provided in the Delegated Admin package at the top level in the file `delegated-admin-account-created.template`. This template provides a multi-part text and HTML email to the user with their user name and initial password, along with a self-service link they can use to log on to PingFederate and change their password and profile information.

ⓘ **Note:** The *PingDirectory*`/webapps/delegator/delegated-admin-account-created.template` template must be copied to the *PingDirectory*`/config/account-status-notification-email-templates/` directory. This file must be copied before running **`delegated-admin.dsconfig`** even if email functionality is not needed, since some commands in dsconfig require this template.

Steps

1. Edit the template as follows:
   a. Uncomment the line that sets the value for `profile_management_url`.
   b. Change the value of `profile_management_url` to the externally accessible URL of the Profile Management Endpoint of your PingFederate Local Identity Profile.

2. Copy the template file to the `config/account-status-notification-email-templates` folder of each instance of PingDirectory Server.

   By default, the email is sent to the address that is found within the user's LDAP mail attribute.

> ⓘ **Note:** A mail value must be provided for each user. For more information, refer to `common-header-fields.vm` in the email templates folder.

Next steps
For more information about the format of the email and further customization, refer to the `README` file in the templates folder.

Creating request criteria to match Delegated Admin user ADD requests

About this task

For each user resource type for which new user email invites need to be sent, create simple request criteria to match the parent DN and object classes for the resource type.

> ⓘ **Note:** The setup script already includes such a request criteria for the user resource type that it creates.

```
$ dsconfig create-request-criteria --criteria-name \
"Delegated Admin User Creation Request Criteria" --type simple \
--set operation-type:add --set \
"included-target-entry-dn:ou=people,dc=example,dc=com" \
--set "any-included-target-entry-filter:(objectClass=inetOrgPerson)" \
--set "included-application-name:PingDirectory Delegated Admin"
```

The included-application-name property ensures that the criteria matches users whom the Delegated Admin created, but not users who were created through another interface, such as the Directory REST API. This application name value is visible in the LDAP access log for operations that the Delegated Admin HTTP servlet invokes.

Creating an SMTP external server

About this task

To send emails, PingDirectory Server must be configured with an SMTP server in the global configuration, as the following example shows.

```
$ dsconfig create-external-server --server-name \
"SMTP Server" --type smtp --set server-host-name:smtp.example.com \
--set user-name:example-smtp-user --set password:example-smtp-password
$ dsconfig set-global-configuration-prop --set \
"smtp-server:SMTP Server"
```

Creating a multi-part Email Account Status notification handler for Delegated Admin user ADD requests

About this task

An Email Account Status Notification Handler must be set in the password policy in force for new users. This handler is typically the Default Password policy.

ⓘ **Note:** The setup script creates an example notification handler in a disabled state. This handler cannot be enabled until an SMTP server becomes available in the global configuration.

The notification handler references the email template in the `config/account-status-notification-email-templates` folder.

Steps

1. Create or enable the handler, as appropriate:
   - To create the handler from scratch, use the **`dsconfig create-account-status-notification-handler`** command, as the following example shows.

     ```
     $ dsconfig create-account-status-notification-handler \
     --handler-name "Delegated Admin Email Account Status \
     Notification Handler" --type multi-part-email --set \
     enabled:true --set \
     "account-creation-notification-request-criteria:Delegated \
     Admin User Creation Request Criteria" --set \
     account-created-message-template:config/account-status-\
     notification-email-templates/delegated-admin-account-created.template
     ```

   - To enable the handler that is provided with the setup script, use the **`dsconfig set-account-status-notification-handler-prop`** command, as the following example shows.

     ```
     $ dsconfig set-account-status-notification-handler-prop \
     --handler-name "Delegated Admin Email Account Status Notification \
     Handler" --set enabled:true
     ```

2. Set the handler in the password policy.

   ```
   $ dsconfig set-password-policy-prop \
   --policy-name "Default Password Policy" --set \
   "account-status-notification-handler:Delegated Admin Email Account \
   Status Notification Handler"
   ```

Enabling the referential integrity plugin

About this task

When a REST resource type is set to create new entries using an RDN attribute other than `entryUUID`, referential integrity issues can result when the entry's RDN attribute is edited. To preserve the referential integrity of group memberships in such a scenario, enable the referential integrity plugin, as follows:

```
bin/dsconfig set-plugin-prop --plugin-name "Referential Integrity" \
  --set enabled:true
```

After the plugin is enabled, group memberships are preserved.

Enable log tracing

Log tracing can be enabled for OAuth token processing, HTTP request and response actions, and API debugging.

To view OAuth token processing and full HTTP request/response tracing, enable the debug trace logger, as follows:

```
$ bin/dsconfig set-log-publisher-prop \
  --publisher-name 'Debug Trace Logger' \
```

```
  --set enabled:true
```

To enable **dadmin** API debug logging, use the following commands:

```
$ bin/dsconfig create-debug-target \
  --publisher-name 'File-Based Debug Logger' \
  --target-name com.unboundid.directory.server.http \
  --set debug-level:VERBOSE
```

```
$ bin/dsconfig create-debug-target \
  --publisher-name 'File-Based Debug Logger' \
  --target-name com.unboundid.directory.server.extensions.dadmin \
  --set debug-level:VERBOSE
```

```
$ bin/dsconfig create-debug-target \
  --publisher-name 'File-Based Debug Logger' \
  --target-name com.unboundid.directory.broker.api \
  --set debug-level:VERBOSE
```

```
$ bin/dsconfig set-log-publisher-prop \
  --publisher-name 'File-Based Debug Logger' \
  --set enabled:true
```

Specify a custom hostname and port for your Directory Server

If this application is being installed on a separate server from your PingDirectory Server instance, you will need to specify where that PingDirectory Server instance can be found within the Delegated Admin application's `config.js` file. The DS_HOST config variable should match your PingDirectory Server's hostname while the DS_PORT config variable should match your PingDirectory Server's HTTPS port.

These do not need to be specified if you are hosting the Delegated Admin application alongside your PingDirectory Server instance as described in previous instructions,.

Change the application logo

To change the Ping logo to your corporate logo, add the following line to the configuration file **config.js**:

```
window.HEADER_BAR_LOGO = '<filename>';
```

Support for corporate logos includes, but is not limited to, the following file types:

- JPG
- PNG
- SVG

Add the logo to the build directory, making certain to use the same file name that you specified earlier. The corporate logo appears in the header, and the Ping logo becomes relegated to the sidebar, in grayscale, with a "Powered by" line above it.

To maintain an appropriate aspect ratio, logo images are resized in Delegated Admin to a height of 22px and a maximum width of 150px.

Configure the session timeout

By default, Delegated Admin features an idle session timeout value of 30 minutes. To adjust this value, perform the following steps:

1. Open the configuration file **config.js** in a text editor.

2. Add the following line, where `{TimeoutValue}` is an integer that represents, in minutes, the session timeout value:

   ```
   window.TIMEOUT_LENGTH_MINS={TimeoutValue};
   ```

   To view an example outline that features this setting, refer to **`example.config.js`**.
3. Save your changes to **`config.js`**.

Verifying the installation

About this task

To verify that Delegated Admin is installed successfully, visit `https://webserverHost:httpPort/delegator` and log on to the application.

If your logon attempt is unsuccessful, see *Enable log tracing* on page 1270, and use your browser's debug feature to gain insight into the token-validation process.

Reporting

Reports provide information about users and group membership. You can preview a report and search, filter, and sort the content. When the preview meets your needs, you can download the resulting report.

In the Delegated Admin GUI, the left navigation pane provides a **Reporting** tab under the top-level sections such as **Users** and **Groups**.

On those **Reporting** tabs, you can:

- Filter the columns (attributes) to display
- Filter the users returned, such as all users with the first name John (exact matches only)
- Sort columns by clicking the column name

To apply filters to the report, click **Run**.

To download the report as currently filtered and sorted, click **Download Report**.

To clear any filtering and sorting, click **Reset**.

Compatibility matrix

The following matrix identifies the minimum versions of PingDirectory Server that are required to access the full functionality provided by each release of Delegated Admin.

All versions of Delegated Admin require PingFederate Server 9.0.0 or later.

Delegated Admin	PingDirectory Server
4.4.1 ⓘ **Note:** This release addresses an issue discussed in *Known issues and limitations* on page 185.	8.2.0.0
4.4.0	8.2.0.0
4.3.0	8.2.0.0 EA

Delegated Admin	PingDirectory Server
4.2.1 ⓘ **Note:** This release addresses a 4.2.0 issue discussed in *Known issues and limitations* on page 185.	8.1.0.0
4.2.0	8.1.0.0
4.1.0	8.1.0.0 EA
4.0.0	8.0.0.0
3.5.1	7.3.0.1
3.5.0	7.3.0.1
3.4.0	7.2.1.0
3.3.0	7.2.1.0
3.2.0	7.2.1.0
3.0.2	7.2.0.1*
3.0.1	7.2.0.0*
3.0.0	7.2.0.0*

*The corresponding version of Delegated Admin is incompatible with version 7.2.1.0 and later.

Configure PingFederate Server

PingFederate offers many configuration options. This appendix provides an example PingFederate configuration that supports Delegated Admin.

ⓘ **Note:** The PingFederate discussions in this appendix are based on the PingFederate 10.1 interface.

Configure PingFederate as the identity provider

This procedure configures PingFederate Server as the identity provider for PingDirectory Server.

Before you begin
Download the LDAPS certificate from PingDirectory Server. For more information, see *Exporting certificates* on page 342.

Steps

1. Sign on to the PingFederate administrative console.
2. Import the PingDirectory Server LDAPS certificate.
   a. Go to **SECURITY**# **Certificate & Key Management**# **Trusted CAs**.
   b. Click **Import**, click **Choose File** to browse to the certificate, click **Next**, and then click **Save**.

3. Add an LDAP datastore.
   a. Go to **SYSTEM**# **Data Stores**.
   b. Click **Add New Data Store**.
   c. Specify a **NAME** for the data store.
   d. Set **TYPE** to **Directory (LDAP)**.
   e. Click **Next**.
   f. In the **Hostname(s)** field, enter the PingDirectory Server host name and LDAPS port, separated by a colon (for example, 10.101.113.75:1389) and click **Add**.
   g. Select the **USE LDAPS** check box.
   h. Set **LDAP TYPE** to **PingDirectory**.
   i. In the **USER DN** field, enter one of the following values based on your PingDirectory configuration.
      - `cn=dmanager`
      - `cn=Directory Manager`
   j. In the **PASSWORD** field, specify the root password.
   k. Click **Advanced** and then **Advanced LDAP Options**.
      1. Select the **CREATE NEW CONNECTIONS IF NECESSARY** check box.
      2. Clear the **VERIFY LDAPS HOSTNAME** check box.
      3. Click **Done**.
   l. Click **Test Connection**.
   m. Click **Next**.
   n. Click **Save**.

4. Create the HTML form IdP Adapter.

   The adapter authenticates users against PingDirectory Server.

   a. Go to **AUTHENTICATION**# **IdP Adapters**# **Create New Instance**
   b. In the **INSTANCE NAME** field, enter a name such as PingDirectoryIdP.
   c. Specify an **INSTANCE ID**.
   d. Set **TYPE** to **HTML Form IdP Adapter**.
   e. Click **Next**.
   f. Go to the bottom of the page and click **Manage Password Credential Validators**.

      Create a validator to authenticate users against PingDirectory Server.

      1. Click **Create New Instance**.
      2. Specify an **INSTANCE NAME**.
      3. Specify an **INSTANCE ID**.
      4. Set **TYPE** to **LDAP User Name Password Credential Validator**.
      5. Click **Next**.
      6. Specify an **LDAP DATASTORE**.
      7. Specify an **SEARCH BASE**.
      8. Enter the following text in the **SEARCH FILTER** field to use the email address or user name to sign on to the system.

         ```
         (|(uid=${username})(mail=${username}))
         ```

      9. Click **Next** and extend the contract with **`entryUUID`** and **`cn`**.

         These values are used later.

      10. Click **Next**, **Done**, or **Save** until you reach the **Create Adapter Instance** screen.

   g. Add a new row to Password Credential Validators, choose the new LDAP Password Credential Validator, and click **Update**.
   h. Go to the **Extended Contract** tab and extend the adapter contract with **`entryUUID`** and **`cn`**.
   i. Go to the **Adapter Attributes** tab, select **`entryUUID`** for a pseudonym, and then click **Next**, **Next**, **Done**, and **Save**.

      For more information, see *Configuring the LDAP Username Password Credential Validator*.

5. Enable session tracking.

   a. Go to **AUTHENTICATION**# **Policies**# **Sessions**
   b. Select the **TRACK ADAPTER SESSIONS FOR LOGOUT** check box.
   c. Select the **TRACK REVOKED SESSIONS ON LOGOUT** check box.
   d. Select the **ENABLE AUTHENTICATION SESSIONS FOR ALL SOURCES** check box.
   e. Click **Save**.

Configure the OAuth server

Steps

1. Sign on to the PingFederate administrative console.

2. Set the IdP Adapter Mapping.
   a. Go to **AUTHENTICATION**# **OAuth**# **IdP Adapter Grant Mapping**.
   b. From the **SOURCE ADAPTER INSTANCE** list, select the IdP Adapter you created in *Configure PingFederate as the identity provider* on page 1273 and click **Add Mapping**.
   c. Click **Next**.

      No attribute source is needed.
   d. On the **Contract Fulfillment** tab, set the contracts as shown in the following table.

Contract	Source	Value
USER_KEY	**Adapter**	**entryUUID**
USER_NAME	**Adapter**	**cn**

   e. Click **Next** and then click **Next** again.
   f. Click **Save**.
3. Set up Access Token Management.

   Select an existing instance or click **APPLICATIONS**# **OAuth**# **Access Token Management**# **Create New Instance**.

   a. If selecting an existing instance, click the **Instance Configuration** tab. (With an existing instance, JSON Web Tokens (JWTs) are configured automatically.).

      If creating a new instance, specify the required fields and set **TYPE** to **JSON Web Tokens**.
   b. Use symmetric encryption for JWT by adding a row in the **Symmetric Keys** section, using 32 bytes or 64 chars of hex.

      This encryption only requires a symmetric key (not a certificate and private key). This step requires the client to validate the token by hitting the validation endpoint on the server.
   c. Set **JWS ALGORITHM** to **HMAC using SHA-256**.
   d. Set **ACTIVE SYMMETRIC KEY ID** to your symmetric key and click **Next**.
   e. On the **Session Validation** tab, select all options and click **Next**.
   f. On the **Access Token Attribute Contract** tab, list at least one attribute to be defined in the access token, add **`sub`**, click **Next** until you reach the last section, and then click **Save**.
4. Set up Access Token Mapping.
   a. Go to **APPLICATIONS**# **OAuth**# **Access Token Mappings**.
   b. Set **CONTEXT** to **Default**, set **ACCESS TOKEN MANAGER** to the Access Token Manager you created in the last step, and click **Add Mapping**.
   c. Click **Next** in the **Attribute Source & User Lookup** section to go to the **Contract Fulfillment** section.
   d. In the **sub** row, make the following selections:
      - From the **Source** list box, select **Persistent Grant**.
      - From the **Value** list box, select **USER_KEY**.
   e. Click **Next** until you reach the **Summary** section, and then click **Save** in the **Summary** section.

5. Set up the OpenID Connect policy.
   a. Go to **APPLICATIONS**# **OAuth**# **OpenID Connect Policy Management**.
   b. Click **Add Policy**.
   c. Specify **POLICY ID**.
   d. Specify **NAME**.
   e. Choose the previously created Access Token Manager and click **Next**.
   f. Delete all extended contract attributes except **`sub`**.
      Other scopes are defined, if configured.
   g. Click **Next** to reach the **Contract Fulfillment** section.
   h. Fulfill the OIDC contract **`sub`** with the Access Token attribute **`sub`**.
   i. Click **Next** and then click **Done**.
   j. If a default OIDC policy is not already defined, set this new policy as the default and click **Save**.
6. Add scopes for PingDirectory Server APIs.
   a. Go to **SYSTEM**# **OAuth Settings**# **Scope Management**.
   b. Click the **Exclusive Scopes** tab.
   c. Add a scope with the value and description given below.

      Scope Value

      ```
      urn:pingidentity:directory-delegated-admin
      ```

      Scope Description

      ```
      DAScope
      ```

   d. Click **Save**.

Configure PingDirectory Server as the token validator (create OAuth client for PingDirectory)

About this task

When creating a Access Token Validator in PingDirectory Server, use the `pingdirectory` client ID and secret. PingDirectory Server uses an identity mapper to match the **`sub`** claim against the **`entryUUID`** attribute.

To configure PingDirectory Server as the token validator, perform the following steps:

Steps

1. Sign on to the PingFederate administrative console.
2. Go to **APPLICATIONS**# **OAuth**# **Clients**.
3. Click **Add Client**.
4. For both the **Client ID** and **Name**, specify `pingdirectory`.
5. In the **CLIENT AUTHENTICATION** section, select **CLIENT SECRET**.
6. In the **CLIENT SECRET** section, select **CHANGE SECRET** and then type or generate a secret.
7. Copy the secret key.
8. In the **ALLOWED GRANT TYPES** section, select **Access Token Validation (Client is a Resource Server)**.
9. Set **DEFAULT ACCESS TOKEN MANAGER** to **Default**.
10. Click **Save**.

Configure Delegated Admin as a new client (create OAuth client for Delegated Admin)

About this task

To configure Delegated Admin as a new client:

Steps

1. Sign on to the PingFederate administrative console.
2. Go to **APPLICATIONS**# **OAuth**# **Clients**.
3. Click **Add Client**.
4. For both the **CLIENT ID** and **NAME**, specify `dadmin`.
5. Set **CLIENT AUTHENTICATION** to **NONE**.

   ⓘ **Note:** Do not set a **CLIENT SECRET**.

6. For **REDIRECT URIS**, specify the URI appropriate for your environment based on the following table and click **Add**.

For Delegated Admin on a PingDirectory server or a PingDirectoryProxy server	`https://<server-host>:1443/delegator/*`
For Delegated Admin on a web server hosted locally	`http://localhost:5000/*`

7. Make the following selections.
   - In the **BYPASS AUTHORIZATION APPROVAL** section, select **Bypass**.
   - In the **EXCLUSIVE SCOPES** section, select **Allow Exclusive Scopes** and then select **urn:pingidentity:directory-delegated-admin**.
   - In the **ALLOWED GRANT TYPES** section, select **Implicit**.
   - For the **DEFAULT ACCESS TOKEN MANAGER**, select the one that was created previously for Delegated Admin.
   - In the **OPENID CONNECT** section, select the OIDC policy that was previously created.
8. Click **Save**.

Next steps

After completing the previous steps, display the logged-in administrator in Delegated Admin and the administrator who generated a report (in the downloaded report).

1. Add the `profile` scope and ensure it is available to the OAuth client used for the Delegated Admin application.
2. Add and fulfill the `name` attribute as part of the contract for both the access token and the ID token that are supplied to the Delegated Admin application.
3. Set the `PROFILE_SCOPE_ENABLED` configuration variable for Delegated Admin in the `config.js` file to `true`.

   For example:

```
/**
 * Configuration wrapper object for Delegated Admin
 */
window.PD_DADMIN_CONFIG = {
  /**
   * Set to true if the "profile" scope is supported for the Delegated
 Admin OIDC client on
```

```
    * PingFederate and you wish to use it to show the current user's name
 in the navigation.
    * DEFAULT: false
    */
   PROFILE_SCOPE_ENABLED: true,
};
```

Set Cross-Origin Resource Sharing (CORS) settings

Steps

1. Sign on to the PingFederate administrative console.
2. Click **SYSTEM**# **OAuth Settings**# **Authorization Server Settings**.
3. Go to the **Cross-Origin Resource Sharing Settings** section.
4. Add "`https://${directoryServer:httpPort}`" to the Allowed origins, using the host name and HTTPS listener port for PingDirectory Server.

   For example, consider the following table.

For Delegated Admin on a PingDirectory server or a PingDirectoryProxy server	`https://<server-host>:1443/delegator/*`
For Delegated Admin on a web server hosted locally	`http://localhost:5000/*`

5. Click **Save**.

Configure PingFederate as a new client (create OAuth client for PingFederate)

About this task

To configure PingFederate as a new client, perform the following steps:

Steps

1. Sign on to the PingFederate administrative console.
2. Go to **APPLICATIONS**# **OAuth**# **Clients**.
3. Click **Add Client**.
4. For both the **CLIENT ID** and **NAME**, specify `pingfederate`.
5. Set **CLIENT AUTHENTICATION** to **CLIENT SECRET**.
6. In the **CLIENT SECRET** section, select **CHANGE SECRET** and then type or generate a secret. This secret must be least 32 characters. You will use it in Delegated Admin setup.
7. For **REDIRECT URIS**, add the following value.

   `https://localhost:8443/client/`
8. Make the following selections.
   - In the **ALLOWED GRANT TYPES** section, select the following items:
     - **Authorization Code**
     - **Implicit**
     - **Refresh Token**
     - **Client Credentials**
     - **Resource Owner Password Credentials**
     - **Access Token Validation (Client is a Resource Server)**
   - Set **DEFAULT ACCESS TOKEN MANAGER** to **RSA256None**.

9. Click **Save**.

Optional configuration tasks

For additional, optional configuration tasks, see the following topics.

- *Defining a local identity profile*
- *Configuring the HTML Form Adapter for customer identities*
- *Configuring local identity mapping*

PingDataSync Administration Guide

PingDirectory™ Product Documentation

Support

https://support.pingidentity.com/

Introduction to PingDataSync Server

PingDataSync Server is a high-capacity, high-reliability data synchronization and transfer pipe between source and destination topologies.

This chapter presents a general overview of the PingDataSync Server process and examples for use.

Topics include:

Overview of PingDataSync Server

Data synchronization process

Synchronization modes

PingDataSync Server operations

Configuration components

Synchronization flow examples

Sample synchronization

Overview of PingDataSync Server

PingDataSync Server is an efficient, Java-based server that provides high throughput, low latency, and bidirectional real-time synchronization between two endpoint topologies consisting of PingDirectory Servers, PingDirectoryProxy Servers, PingOne, and/or Relational Database Management Systems (RDBMS) systems. PingDataSync Server uses a dataless approach that synchronizes changes directly from the data sources in the background so that applications can continue to update their data sources directly. PingDataSync Server does not store any data from the endpoints themselves, thereby reducing hardware and administration costs. The server's high-availability mechanisms also makes it easy to fail over from the main PingDataSync Server to redundant instances.

Designed to run with little administrative maintenance, PingDataSync Server includes the following features:

- High performance and availability with built-in redundancy.
- Dataless virtual architecture for a small-memory footprint and easy maintenance.
- Hassle-free setup that enables mapping attribute names, values, and DNs between endpoints. For directory server endpoints, this enables making schema and Directory Information Tree (DIT) changes without custom coding and scripting.
- Multi-vendor directory server support including the PingDirectory Server, PingDirectoryProxy Server, Nokia 8661 Directory Server, Nokia 8661 Directory Proxy Server, Oracle/Sun Directory Server Enterprise Edition, Oracle/Sun Directory Server, Oracle Unified Directory, OpenDJ, and Microsoft Active Directory, and generic LDAP directories.
- RDBMS support including Oracle Database, and Microsoft SQL Server systems.
- Proxy Server support including the PingDirectoryProxy Server and the Nokia 8661 Directory Proxy Server.
- Notification support that allows real-time change notifications to be pushed to client applications or services as they occur.

Data synchronization process

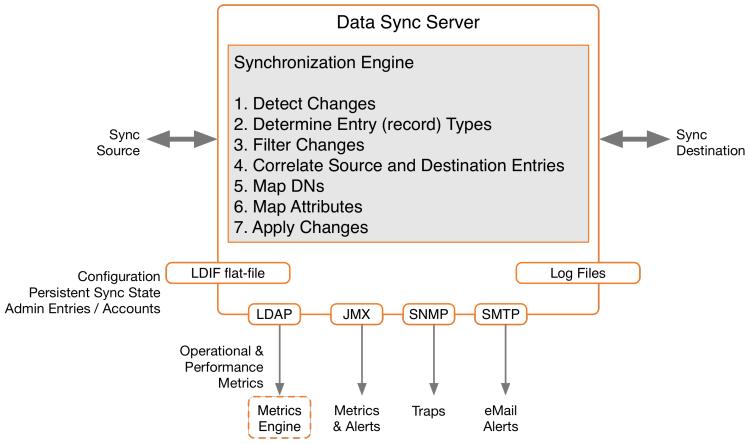
PingDataSync Server performs point-to-point synchronization between a source endpoint and a destination endpoint. An endpoint is defined as any source or destination topology of directory or database servers.

PingDataSync Server synchronizes data in one direction or bidirectionally between endpoints.
For example, in a migration phase from Sun Directory Server to a PingData PingDirectory Server, synchronization can occur in one direction from the source server to a staging server. With one-way synchronization, the source server is the authoritative endpoint for changes in the system. Bidirectional synchronization allows for parallel active installations between the source and the destination endpoints. With bidirectional synchronization, both endpoints are authoritative for the same set of attributes or for different sets of data.

PingDataSync Server also contains no single point of failure, either for detecting changes or for applying changes. PingDataSync Server instances themselves are redundant. There can be multiple instances running at a time, but only the server with the highest priority is actively synchronizing changes. The stand-by servers are constantly polling the active server instance to update their persistent state. This state contains the minimum amount of information needed to begin synchronization where the primary server left off, which logically is the last processed change number for the source server. In the case of a network partition, multiple servers can synchronize simultaneously without causing problems as they each verify the full entry before making changes.

Synchronization architecture

PingDataSync Server uses a virtualized, dataless approach that does not store directory data locally. The log files, administrator entries, configuration, sync state information are stored as flat files (LDIF format) within the system. No additional database is required.



Change tracking, monitoring, and logging

PingDataSync Server tracks and manages processes and server health with the following tools:

Change Tracking

Each directory instance stores a separate entry under cn=changelog for every modification made to the directory. PingDataSync Server provides full control over the synchronization process by determining which entries are synchronized, how they are correlated to the entries at the destination endpoint, and how they are transformed into the destination schema.

- For the PingData PingDirectory Server or Nokia 8661 Directory Server topologies, PingDataSync Server uses the server's LDAP Change Log for modification detection.
- For Oracle/Sun Directory Server, OpenDJ, Oracle Unified Directory, and generic LDAP directory topologies, PingDataSync Server uses the server's Retro Change Log, which provides a detailed summary of each change.
- For Active Directory, PingDataSync Server uses the DirSync control, which polls for object attribute changes.
- For RDBMS systems, PingDataSync Server uses an Ping Identity Server SDK plugin to interface with a customized RDBMS change log table. Database triggers on each table record all INSERT, UPDATE, and DELETE operations to the change log table.

Monitoring, Alerts, and Alarms

PingDataSync Server supports several industry-standard, administrative protocols for monitoring, alarms, and alerts. System alarms and gauges can be configured to determine healthy performance thresholds and the server actions taken when performance values are outside the threshold.
All administrative alarms are exposed over LDAP as entries under base DN `cn=alarms`.
An administrative alert framework sends warnings, errors, or other server events through log messages, email, or JMX notifications. Administrative alerts are also exposed over LDAP as entries below base DN `cn=alerts`. Typical alert events are startup or shutdown, applied configuration changes, or synchronized resources unavailable.

Logging

PingDataSync Server provides standard logs (`sync, access, error, failed-operations, config-audit.log, debug`). The server can also be configured for multiple active sync logs. For

example, each detected change, each dropped change, each applied change, or each failed change can be logged.

Synchronization modes

PingDataSync Server runs as a standalone Java process with two synchronization modes: standard and notification.

Standard synchronization

In standard synchronization mode, PingDataSync Server polls the directory server change log for create, modify, and delete operations on any entry. The server fetches the full entries from both the source and destination endpoints, and compares them to produce the minimal set of changes required to synchronize the destination with the source.

The following shows the standard synchronization change flow between two servers. The changes are processed in parallel, which increases throughput and offsets network latency.

Notification synchronization

In notification synchronization mode, PingDataSync Server skips the fetch and compare phases of processing and simply notifies the destination that a change has happened and provides the details of the change. Notification mode is currently available for the PingData and Alcatel-Lucent 8661 directory and proxy servers only.

PingDataSync Server operations

PingDataSync Server provides seamless integration between disparate systems to transform data using attribute and DN mappings. A bulk resynchronization operation can be run verify mappings and test synchronization settings.

Real-time synchronization

Real-time synchronization is performed with the `realtime-sync` utility. The `realtime-sync` utility polls the source server for changes and synchronizes the destination entries immediately. Once the server determines that a change should be synchronized, it fetches the full entry from the source. It then searches for the corresponding entry in the destination endpoint using correlation rules and applies the minimum set of changes to synchronize the attributes. The server fetches and compares the full entries to make sure it does not synchronize any old data from the change log.

After a synchronization topology is configured, run `resync` to synchronize the endpoints, and then run `realtime-sync` to start global synchronization.

The `realtime-sync` tool is used for the following tasks:

- Start or stop synchronization globally or for specific sync pipes only.
- Set a start point at which synchronization should begin such as the beginning or end of the change log, at a specified change number, at a specified change sequence number, or at a specified time frame in the change log.

Data transformations

Data transformations alter the contents of synchronized entries between the source and destination directory server to handle variances in attribute names, attribute values, or DN structures. When entries are synchronized between a source and a destination server, the contents of these entries can be changed using attribute and DN mappings, so that neither server needs be aware of the transformations.

- Attribute Mapping – Any attribute in the entry can be renamed to fit the schema definitions from the source endpoint to the destination endpoint. This mapping makes it possible to synchronize information stored in one directory's attribute to an attribute with a different name in another directory server, or to construct an attribute using portions of the source attribute values.

- DN Mapping – Any DNs referenced in the entries can be transparently altered. This mapping makes it possible to synchronize data from a topology that uses one DIT structure to a system that uses a different DIT structure.

Bulk resync

The `resync` tool performs a bulk comparison of entries on source topologies and destination topologies. PingDataSync Server streams entries from the source, and either updates the corresponding destination entries or reports those that are different. The `resync` utility resides in the `/bin` folder (UNIX or LINUX) or `\bat` folder (Windows), and can be used for the following tasks:

- Verify that the two endpoints are synchronized after an initial configuration.
- Initially populate a newly configured target endpoint.
- Validate that the server is behaving as expected. The `resync` tool has a `--dry-run` option that validates that synchronization is operating properly, without updating any entries. This option also can be used to check attribute or DN mappings.
- Perform scheduled synchronization.
- Recover from a failover by resynchronizing entries that were modified since the last backup was taken.

The `resync` tool also enables control over what can be synchronized, such as:

- Include or exclude any source and destination attributes.
- Apply an LDAP filter to only sync entries created since that last time the tool ran.
- Synchronize only creations or only modifications.
- Change the logging verbosity.
- Set a limit on `resync` operations (such as 2000 operations per second) to reduce impact on endpoint servers.

The sync retry mechanism

PingDataSync Server is designed to quickly synchronize data and attempt a retry should an operation fail for any reason. The retry mechanism involves two possible retry levels, which are configurable on the Sync Pipe configuration using advanced Sync Pipe properties. For detailed information, see the *PingDataSync Server Reference Guide* for the Sync Pipe configuration parameters.

Retry involves two possible levels:

First Level Retry

If an operation fails to synchronize, the server will attempt a configurable number of retries. The total number of retry attempts is set in the `max-operation-attempts` property on the Sync Pipe. The property indicates how many times a worker thread should retry the operation before putting the operation into the second level of retry, or failing the operation altogether.

Second Level Retry

Once the `max-operation-attempts` property has been exceeded, the retry is sent to the second level, called the delayed-retry queue. The delayed-retry queue uses two advanced Sync Pipe properties to determine the number of times an operation should be retried in the background after a specified delay.

Operations that make it to this level will be retried after the `failed-op-background-retrydelay` property (default: 1 minute). Next, PingDataSync Server checks the `max-failedop-background-retries` property to determine the number of times a failed operation should be retried in the background. By default, this property is set to 0, which indicates that no background retry should be attempted, and that the operation should be logged as failed.

ⓘ **Note:**

Background operations can hold up processing other changes, since PingDataSync Server will only process up to the next 5000 changes while waiting for a retried operation to complete.

Retry can be controlled by the custom endpoint based on the type of error exception. When throwing an exception, the endpoint code can signal that a change should be aborted, retried a limited number of times, or retried an unlimited number of times. Some errors, such as endpoint server down, should be retried indefinitely.

If the `max-failed-op-background-retries` property has been exceeded, the retry is logged as a failure and appears in the sync and the sync-failed-ops logs.

Configuration components

PingDataSync Server supports the following configuration parameters that determine how synchronization takes place between directories or databases:

Sync Pipe

Defines a single synchronization path between the source and destination topologies. Every Sync Pipe has one or more Sync Classes that control how and what is synchronized. Multiple Sync Pipes can run in a single server instance.

Sync Source

Defines the directory topology that is the source of the data to be synchronized. A Sync Source can reference one or more supported external servers.

Sync Destination

Defines the topology of directory servers where changes detected at the Sync Source are applied. A Sync Destination can reference one or more external servers.

External Server

Defines a single server in a topology of identical, replicated servers to be synchronized. A single external server configuration object can be referenced by multiple Sync Sources and Sync Destinations

Sync Class

Defines the operation types and attributes that are synchronized, how attributes and DNs are mapped, and how source and destination entries are correlated. A source entry is in one Sync Class and is determined by a base DN and LDAP filters. A Sync Class can reference zero or more Attribute Maps and DN Maps, respectively. Within a Sync Pipe, a Sync Class is defined for each type of entry that needs to be treated differently. For example, entries that define attribute mappings, or entries that should not be synchronized at all. A Sync Pipe must have at least one Sync Class but can refer to multiple Sync Class objects.

DN Map

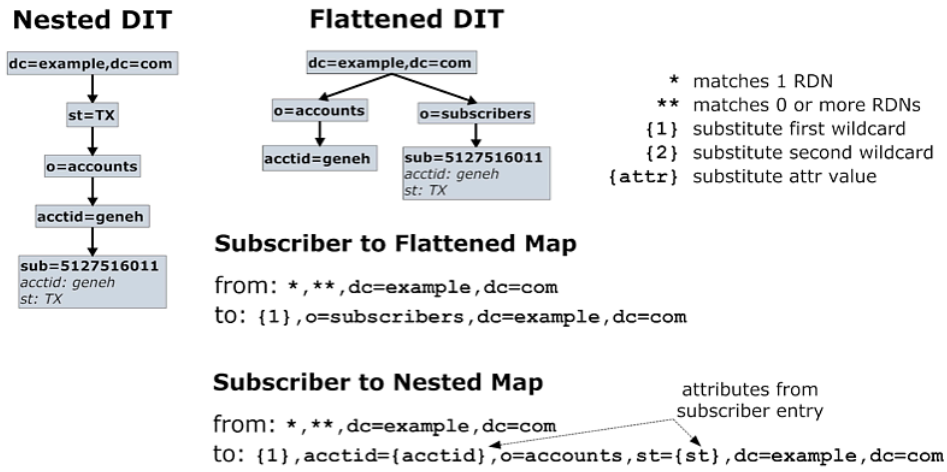
Defines mappings for use when destination DNs differ from source DNs. These mappings allow the use of wild cards for DN transformations. A single wild card ("*") matches a single RDN component and can be used any number of times. The double wild card ("**") matches zero or more RDN components and can be used only once. The wild card values can be used in the `to-dn-pattern` attribute using `{1}` and their original index position in the pattern, or `{attr}` to match an attribute value. For example:

```
**,dc=myexample,dc=com->{1},o=example
```

Regular expressions and attributes from the user entry can also be used. For example, the following mapping constructs a value for the `uid` attribute, which is the RDN, out of the initials (first letter of `givenname` and `sn`) and the employee ID (`eid` attribute).

```
uid={givenname:/^(.)(.*)/$1/s}{sn:/^(.)(.*)/$1/s}{eid},{2},o=example
```

The following illustrates a how a nested DIT can be mapped to a flattened structure.



Attribute Map and Attribute Mappings

Defines a mapping for use when destination attributes differ from source attributes. A Sync Class can reference multiple attribute maps. Multiple Sync Classes can share the same attribute map. There are three types of attribute mappings:

- Direct Mapping – Attributes are directly mapped to another attribute: For example:

  ```
  employeenumber->employeeid
  ```

- Constructed Mapping – Destination attribute values are derived from source attribute values and static text. For example:

  ```
  {givenname}.{sn}@example.com->mail
  ```

- DN Mapping – Attributes are mapped for DN attributes. The same DN mappings that map entry DNs can be referenced. For example:

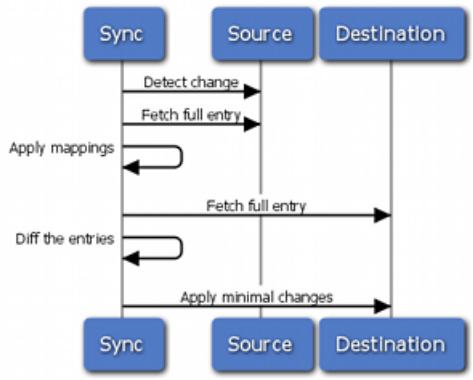
  ```
  uid=jdoe,ou=People,dc=example,dc=com.
  ```

Sync flow examples

PingDataSync Server processes changes by fetching the most up-to-date, full entries from both sides and then compares them. This process flow is called standard synchronization mode. The processing flow differs depending on the type of PingDataSync Server change (`ADD, MODIFY, DELETE, MODDN`) that is requested. The following examples show the control flow diagrams for the sync operations, especially for those cases when a `MODIFY` or a `DELETE` operation is dropped. The sync log records all completed and failed operations.

Modify operation example

About this task

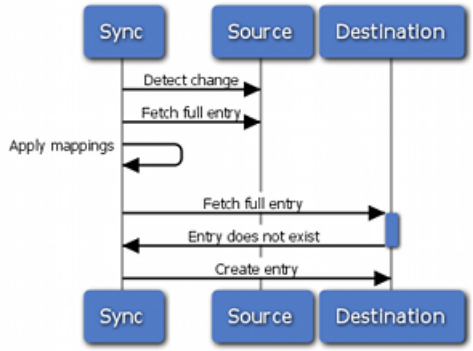


Steps

1. Detect change from the change log table on the source.
2. Fetch the entry or table rows from affected tables on the source.
3. Perform any mappings and compute the equivalent destination entry by constructing an equivalent LDAP entry or equivalent table row.
4. Fetch the entry or table rows from affected tables on the destination.
5. Diff the computed destination entry and actual destination entry.
6. Apply the changes to the destination.

Add operation example

About this task

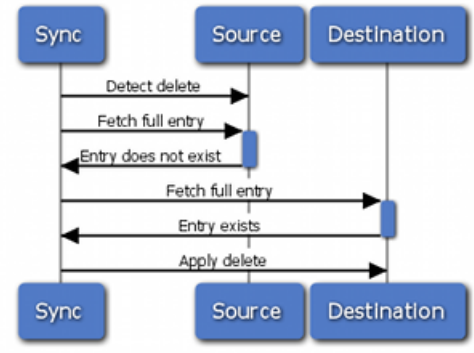


Steps

1. Detect change from the change log table on the source.
2. Fetch the entry or table rows from affected tables on the source.
3. Perform any mappings and compute the equivalent destination entry by constructing an equivalent LDAP entry or equivalent table row.
4. Fetch the entry or table rows from affected tables on the destination.
5. The entry or table row does not exist on the destination.
6. Create the entry or table row.

Delete operation example

About this task

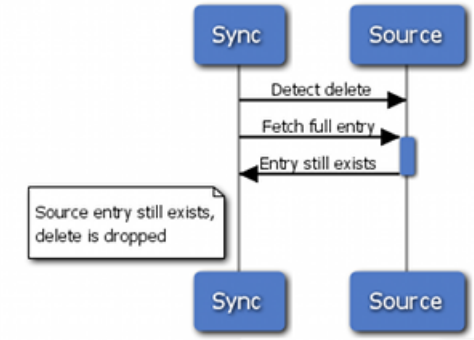


Steps

1. Detect delete from the change log table on the source.
2. Fetch the entry or table rows from affected tables on the source.
3. Perform any mappings and compute the equivalent destination entry by constructing an equivalent LDAP entry or equivalent table row.
4. Fetch the entry or table rows from affected tables on the destination.
5. The entry or table row exists on the destination.
6. Apply the delete on the destination.

Delete after source entry is re-added

About this task

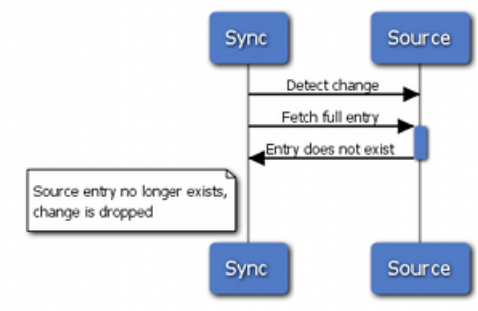


Steps

1. Detect delete from the change log table on the source.
2. Fetch the entry or table rows from affected tables on the source.
3. The entry or table row exists on the source.
4. Delete request is dropped.

Standard modify after source entry is deleted

About this task

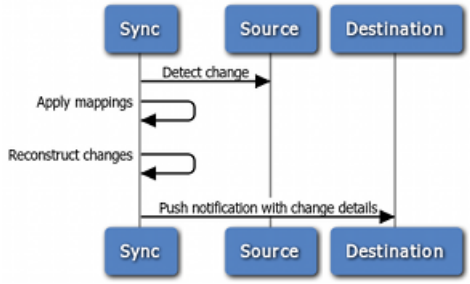


Steps

1. Detect change from the change log table on the source.
2. Fetch the entry or table rows from affected tables on the source.
3. The entry does not exist.
4. Change request is dropped because the source entry no longer exists.

Notification add, modify, modifyDN, and delete

About this task



Steps

1. Detect change from the change log table on the source.
2. Perform any mappings and compute the equivalent destination entry by constructing an equivalent LDAP entry or equivalent table row.
3. Reconstruct changed entries.
4. Push notification with change details to the destination.

Sample synchronization

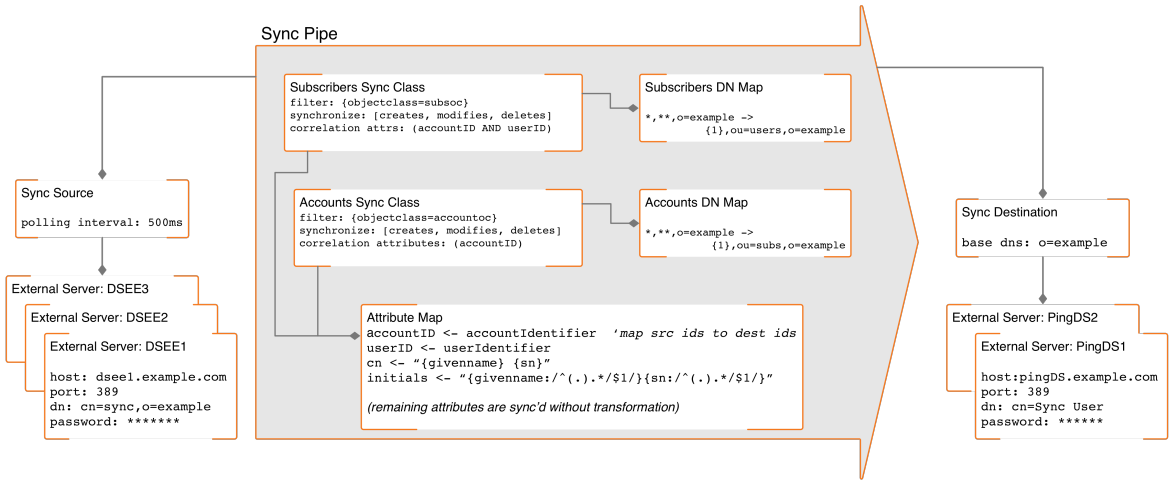
The following is a synchronization migration example from a Sun Directory Server Enterprise Edition (DSEE) topology to the PingData PingDirectory Server topology, including a change in the DIT structure to a flattened directory structure. The Sync Pipe connects the Sun Directory Server topology as the Sync Source and the PingDirectory Server topology as the Sync Destination. Each endpoint is defined with three external servers in their respective topology. The Sync Pipe destination has its base DN set to `o=example`, which is used when performing LDAP searches for entries.

Two Sync Classes are defined: one for Subscribers, and one for Accounts. Each Sync Class uses a single "Sun DS to PingData Attribute Map" that has four attribute mappings defined. Each Sync Class also defines its own DN maps. For example, the Accounts Sync Class uses a DN map, called PingData

Account Map, that is used to flatten a hierarchical DIT, so that the Account entries appear directly under `ou=accounts`as follows:

```
*,**,o=example ->{1},ou=accounts,o=example
```

With this mapping, if an entry DN has `uid=jsmith,ou=people,o=example`, then "*" matches `uid=jsmith`, "**" matches `ou=people`, and `{1}` matches `uid=jsmith`. Therefore, `uid=jsmith,ou=people,o=example`gets mapped to `uid=jsmith,ou=accounts,o=example`. A similar map is configured for the Subscribers Sync Class.



Installing PingDataSync Server

This section describes how to install and run PingDataSync Server. It includes pre- installation requirements and considerations.

Topics include:

System requirements on page 237

Install the JDK on page 1292

Optimize the Linux operating system on page 1292

Ping license keys on page 261

Installing PingDataSync Server on page 1297

Sign on to the Administrative Console on page 246

Server folders and files on page 1299

Start and stop the server on page 1299

Run the server as a Microsoft Windows service on page 1300

Uninstall the server on page 1301

Update servers in a topology on page 1302

Revert an update on page 1304

Install a failover server on page 1305

Administrative accounts on page 1306

System requirements

The following section details system requirements that are qualified and certified to be compatible with the PingDataSync Server.

Differences in operating system versions, service packs, and other platform variations are supported until the platform or other required software are suspected of causing issues.

Platforms

The following operating systems support a PingDirectory Server installation and deployment.

- Windows Server 2019
- Windows Server 2016
- Red Hat Enterprise Linux ES 8.1
- Red Hat Enterprise Linux ES 7.7
- CentOS 8.1
- CentOS 7.7
- SUSE Linux Enterprise 15 SP1
- SUSE Linux Enterprise 12 SP5
- Ubuntu 18.04 LTS
- Ubuntu 16.04 LTS
- Amazon Linux 2

> ⓘ **Attention:**
>
> This product has been tested with the default configurations of all operating system components. Customized implementations or installed third-party plugins from your organization might affect the deployment of this product.

Docker

The following Docker version and operating systems are a pre-installation requirement.

- Version: Docker 19.03.5
- Host operating system: Ubuntu 18.04 LTS
- Base image operating system: Ubuntu 18.04 LTS
- Base image operating system: Alpine Linux 3.11

For more information, view the PingDataSync Server Docker Image on DockerHub at *pingidentity/pingdirectory* and visit the *PingIdentity DevOps documentation*.

> ⓘ **Note:**
>
> Only the PingDataSync Server software is licensed under Ping Identity's end user license agreement, and any other software components contained within the image are licensed solely under the terms of the applicable open source/third-party license.
>
> Ping Identity accepts no responsibility for the performance of any specific virtualization software and in no way guarantees the performance or interoperability of any virtualization software with its products.

Java Runtime Environment

The following Java Development Kit versions are a pre-installation requirement.

- Oracle JDK 8 64-bit
- Oracle JDK 11 LTS 64-bit
- OpenJDK 8 64-bit
- OpenJDK 11
- Corretto 8

- Corretto 11

> ⓘ **Note:**
>
> The *Ping Identity Java Support Policy* applies to your Java Runtime Environment.

Browsers

Administration Console

- Chrome
- Firefox
- Internet Explorer 11 or later

End users

- Chrome
- Edge
- Firefox
- Internet Explorer 11 or later
- Safari

Upgrade overview and considerations

Considerations when upgrading to 8.2.0.0

> ⓘ **Important:**
>
> If you plan to upgrade servers using a mixed-version environment where one version is earlier than 7.0 and some of the servers are still using the admin backend while others have been updated to the topology registry, do not attempt to make size changes to the topology. You cannot remove any existing servers (using `dsreplication disable`) or add new servers (using `dsreplication enable`) when in this transitional state of partially-updated servers. When a topology has been completely migrated to a 7.0 or later version with the topology registry, changes to the topology size are allowed, even in mixed-version environments (for example, mixed 7.3 and 8.3).

This upgrade moves to Jetty 9.4. As a result, the HTTPS connection handler will no longer support TLS_RSA ciphers by default. If you use any legacy HTTPS clients that still require TLS_RSA ciphers, modify the `ssl-cipher-suite` property of the HTTPS Connection Handler to include them.

Install the JDK

The Java 64-bit JDK is required on the server. Even if Java is already installed, create a separate Java installation for use by the server to ensure that updates to the system-wide Java installation do not inadvertently impact the installation.

Optimize the Linux operating system

Configure the Linux file system by making the following changes.

> ⓘ **Note:**
>
> The server explicitly overrides environment variables like `PATH`, `LD_LIBRARY_PATH`, and `LD_PRELOAD` to ensure that settings used to start the server do not inadvertently impact its behavior. If these variables

must be edited, set values by editing the `set_environment_vars` function of the `lib/_script-util.sh` script. Stop and restart the server for the change to take effect.

Setting the file descriptor limit

About this task

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

ⓘ **Note:**

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

Steps

1. Display the current `fs.file-max` limit of the system.

   ```
   sysctl fs.file-max
   ```

   The `fs.file-max` limit is the maximum server-wide file limit you can set without tuning the kernel parameters in the `proc` file system.
2. Edit the `/etc/sysctl.conf` file. If there is a line that sets the value of the `fs.file-max` property, make sure that its value is set to at least 1.5 times the per-process limit. If there is no line that sets a value for this property, add the following to the end of the file (100000 is just an example here; specify a value of at least 1.5 times the per-process limit):

   ```
   fs.file-max = 100000
   ```
3. Display the current hard limit of the system.

   ```
   ulimit -aH
   ```

   The `open files (-n)` value is the maximum number of open files per process limit.

   Verify that its value is set to at least 65535.
4. Edit the `/etc/security/limits.conf` file. If the file has lines that set the soft and hard limits for the number of file descriptors, make sure the values are set to 65535. If the lines are not present, add the following lines to the end of the file (before “#End of file”), making certain to insert a tab between the columns.

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

   ⓘ **Note:**

   The number of open file descriptors is limited by the physical memory available to the host. You can determine this limit with the following command.

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

   If the `file-max` value is significantly higher than the 65535 limit, consider increasing the file descriptor limit to between 10% and 15% of the system-wide file descriptor limit. For example, if the `file-max`

value is 810752, you could set the file descriptor limit to 100000. If the `file-max` value is lower than 65535, the host is likely not sized appropriately.

5. Reboot the system, and then use the `ulimit` command to verify that the file descriptor limit is set to `65535` with the following command:

```
ulimit -n
```

Next steps

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

ⓘ **Note:**

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

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

Add or edit the following lines if they do not already exist:
*      soft    nproc     65536
*      soft    nofile    65536
*      hard    nproc     65536
*      hard    nofile    65536
root   soft    nproc     unlimited
```

Set the file system flushes

Linux systems running the ext3 file system only flush data to disk every five seconds. If the server is on a Linux system, edit the mount options to include the following:

```
commit=1
```

This variable changes the flush frequency from five seconds to one. Also, set the flush frequency in the `/etc/fstab` file to make sure the configuration remains after reboot.

Install sysstat and pstack on Red Hat

The server troubleshooting tool `collect-support-data` relies on the `iostat`, `mpstat`, and `pstack` utilities to collect monitoring, performance statistics, and stack trace information on the server's processes. For Red Hat systems, make sure that these packages are installed, for example:

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

Install the dstat utility

The `dstat` utility is used by the `collect-support-data` tool.

Disable file system swapping

It is recommended that any performance tuning services like `tuned` be disabled. As root, change the current value in the operating system and by adding a line `vm.swappiness = 0` to `/etc/sysctl.conf` to ensure that the correct setting is applied when the system restarts.

If performance tuning is required, `vm.swappiness` can be set by cloning the existing performance profile then adding `vm.swappiness = 0` to the new profile's `tuned.conf` file in `/usr/lib/tuned/profile-name/tuned.conf`. The updated profile is then selected by running `tuned-adm profile customized_profile`.

Manage system entropy

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

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

Set file system event monitoring (inotify)

An event monitoring tool such as inotify can be configured for notifying processes about file system events (including file creation, deletion, and updates). The Linux system puts a limit on the number of inotify watches assigned to each user. To increase the limit, edit `etc/sysctl.conf` to add a line:

```
fs.inotify.max_user_watches =524288
```

Run the command:

```
$ sudo sysctl -wfs.inotify.max_user_watches=524288
```

Tune IO scheduler

Using the correct IO scheduler can increase performance and reduce the possibility of database timeouts when the system is under extreme write load. For file systems running on an SSD, or in a virtualized environment, the `noop` scheduler is recommended. For all other systems, the `deadline` scheduler is recommended. To determine which scheduler is configured on your system, run this command:

```
$ cat /sys/block/<block-device>/queue/scheduler
```

For example:

```
$ cat /sys/block/sda/queue/scheduler
```

Changing the scheduler on a running system can be done with the following command:

```
$ echo 'deadline' > /sys/block/sda/queue/scheduler
```

The change will take effect after the system is restarted. The procedure for configuring a scheduler to use at startup depends on the version of Linux. See the Linux documentation for your specific version for the correct way to configure this setting.

Enable the server to listen on privileged ports

Linux provides 'capabilities' used to grant specific commands the ability to do things that are normally only allowed for a root account. Instead of granting the ability to a specific user, capabilities are granted to a specific command. It may be convenient to enable the server to listen on privileged ports while running as a non-root user.

The `setcap` command is used to assign capabilities to an application. The `cap_net_bind_service` capability enables a service to bind a socket to privileged ports (port numbers less than 1024). If Java

is installed in `/ds/java` (and the Java command to run the server is `/ds/java/bin/java`), the Java binary can be granted the `cap_net_bind_service` capability with the following command:

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

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

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

Run the following command for the change to take effect:

```
$ sudo ldconfig -v
```

Ping license keys

License keys are required to install, update, and renew all Ping products.

How to obtain a license

To obtain a license key, contact your account representative or use the *Ping Identity licensing portal*.

When do you need a license

A license is required for setting up a new single server instance and can be used site-wide for all servers in an environment. Additionally, you must obtain a new license when updating a server to a new major version, such as when upgrading from 7.3 to 8.0. When cloning a server instance with a valid license, you do not need a new license.

> ⓘ **Note:**
>
> The update process displays a prompt for a new license.

How to specify a license

- Specify a license at setup

  You have these options:

  - Use the `--licenseKeyFile` *<path-to-license>* option with **`setup`**.
  - Copy the license file to the server root directory and then run the **`setup`** tool. The tool discovers the license file.
- Specify a license after setup

  Use the Administrative Console or **`dsconfig`** (in the Topology section, select License).

  > ⓘ **Note:** Placing the new license file in the server root directory does not work in this case.

How to view the license status

To view the details of a license, including its expiration, you have these options:

- The server's **`status`** tool
- The Administrative Console's **Status** page (On the **Monitors** tab, search for License.)

License expiration

The server provides a notification as the expiration date approaches.

Before a license expires, obtain a new one and install it by using **`dsconfig`** or the Administrative Console.

> ⓘ **Note:**
>
> An expiring license causes alerts and alarms but does not affect the functionality of the product.

Installing PingDataSync Server

About this task

Use the `setup` tool to install the server. The server needs to be started and stopped by the user who installed it.

> ⓘ **Note:**
>
> A Windows installation requires that the Visual Studio 2010 runtime patch be installed prior to running the `setup` command.

Steps

1. Log in as a user, other than root.
2. Obtain the latest zip release bundle from Ping Identity and unpack it in a directory owned by this user.

   ```
   $ unzip PingDataSync-<version>.zip
   ```
3. Change to the server root directory.

   ```
   $ cd PingDataSync
   ```
4. Run the `setup` command.

   ```
   $ ./setup
   ```
5. Type `yes` to accept the End-User License Agreement and press Enter to continue.
6. If adding this server to an existing PingDataSync Server topology, type `yes`, or press Enter to accept the default (no).
7. Enter the fully qualified host name or IP address of the local host.
8. Create the initial root user DN for PingDataSync Server, or press Enter to accept the default (cn=Directory Manager).
9. Enter and confirm a password for this account.
10. Press Enter to enable server services and the Administrative Console.
11. Enter the port on which PingDataSync Server will accept connections from HTTPS clients, or press Enter to accept the default.
12. Enter the port on which PingDataSync Server will accept connections from LDAP clients, or press Enter to accept the default.
13. Press Enter to enable LDAPS, or enter no.
14. Press Enter to enable StartTLS, or enter no.
15. Select the certificate option for this server.

16. Choose the desired encryption for the directory data, backups, and log files from the choices provided:

- Encrypt data with a key generated from an interactively provided passphrase. Using a passphrase (obtained interactively or read from a file) is the recommended approach for new deployments, and you should use the same encryption passphrase when setting up each server in the topology.
- Encrypt data with a key generated from a passphrase read from a file.
- Encrypt data with a randomly generated key. This option is primarily intended for testing purposes, especially when only testing with a single instance, or if you intend to import the resulting encryption settings definition into other instances in the topology.
- Encrypt data with an imported encryption settings definition. This option is recommended if you are adding a new instance to an existing topology that has older server instances with data encryption enabled.
- Do not encrypt server data.

17. Choose the option for the amount of memory that should be allocated to the server.

18. To start the server when the configuration is complete, press Enter for (yes).

19. A Setup Summary is displayed. choose the option to setup the server with the listed parameters, change the parameters, or cancel the setup.

Next steps

After the server configuration is complete, the `create-sync-pipe-config` tool can be run to configure the synchronization environment.

The PingDataSync Server Administrative Console enables browser-based server management, the `dsconfig` tool enables command line management, and the Configuration API enables management by third-party interfaces.

Sign on to the Administrative Console

After the server is installed, access the Administrative Console, `https://<host>/console/login`, to verify the configuration and manage the server. The root user DN or the common name of a root user DN is required to log into the Administrative Console. For example, if the DN created when the server was installed is `cn=Directory Manager`, `directory manager`can be used to log into the Administrative Console.

If the Administrative Console needs to run in an external container, such as Tomcat, a separate package can be installed according to that container's documentation. Contact Ping Identity Customer Support for the package location and instructions.

ⓘ **Note:** The session timeout for the console is 24 hours by default. When this duration is exceeded, all inactive users are logged off automatically.

To set a different timeout value, configure the `server.sessionTimeout` application parameter, which specifies the timeout duration in seconds. You can set the value as an init parameter either in the console or on the command line, as shown below.

- Console

  Select **Web Application Extensions**# **Console** and then use the **Init Parameter** field.
- Command line

  The following example uses a value of 1800 seconds (30 minutes).

```
dsconfig set-web-application-extension-prop --no-prompt \
--extension-name Console \
--add init-parameter:server.sessionTimeout=1800
```

For the change to take effect, restart the HTTP(S) Connection Handler or the server itself.

Server folders and files

After the distribution file is unzipped, the following folders and command-line utilities are available:

Directories/Files/Tools	Description
ldif	Stores any LDIF files that you may have created or imported.
import-tmp	Stores temporary imported items.
classes	Stores any external classes for server extensions.
bak	Stores the physical backup files used with the backup command-line tool.
velocity	Stores Velocity templates that define the server's application pages.
update.bat, and update	The update tool for UNIX/Linux systems and Windows systems.
uninstall.bat, and uninstall	The uninstall tool for UNIX/Linux systems and Windows systems.
ping_logo.png	The image file for the Ping Identity logo.
setup.bat, and setup	The setup tool for UNIX/Linux systems and Windows systems.
revert-update.bat, and revert-update	The revert-update tool for UNIX/Linux systems and Windows systems.
README	README file that describes the steps to set up and start the server.
License.txt	Licensing agreement for the product.
legal-notices	Stores any legal notices for dependent software used with the product.
docs	Provides the release notes, Configuration Reference (HTML), API Reference, and all other product documentation.
metrics	Stores the metrics that can be gathered for this server and surfaced in the PingDataMetrics Server.
bin	Stores UNIX/Linux-based command-line tools.
bat	Stores Windows-based command-line tools.
lib	Stores any scripts, jar files, and library files needed for the server and its extensions.
collector	Used by the server to make monitored statistics available to the PingDataMetrics Server.
locks	Stores any lock files in the backends.
tmp	Stores temporary files.
resource	Stores the MIB files for SNMP and can include ldif files, make-ldif templates, schema files, dsconfig batch files, and other items for configuring or managing the server.
config	Stores the configuration files for the backends (admin, config) as well as the directories for messages, schema, tools, and updates.
logs	Stores log files.
AD-Password-Sync-Agent.zip	The Active Directory Sync Agent package.

Start and stop the server

To start PingDataSync Server, run the `bin/start-sync-server` command on UNIX or Linux systems (the `bat` folder on Microsoft Windows systems).

Start the server as a background process

Navigate to the server root directory, and run the following command:

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

For Windows systems:

```
$ bat/start-server
```

Start the server at boot time

About this task

By default, the server does not start automatically when the system is booted. To configure the server to start automatically, use the `create-rc-script` tool to create a run control script as follows:

Steps

1. Create the startup script. In this example `ds` is the user.

   ```
   $ bin/create-rc-script \
     --outputFile Ping-Identity-Sync.sh \
     --userName ds
   ```

2. Log in as root, move the generated `Ping-Identity-Sync.sh` script into the `/etc/init.d` directory, and create symlinks to it from the `/etc/rc3.d` (starting with an "S" to start the server) and `/etc/rc0.d` directory(starting with a "K" to stop the server).

   ```
   # mv Ping-Identity-Sync.sh /etc/init.d/
   # ln -s /etc/init.d/Ping-Identity-Sync.sh /etc/rc3.d/S50-Ping-
   IdentitySync.sh
   # ln -s /etc/init.d/Ping-Identity-Sync.sh /etc/rc0.d/K50-Ping-
   IdentitySync.sh
   ```

Stop the server

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

A configuration option can also be set that specifies the maximum shutdown time a process can take.

To stop the server, navigate to the server root directory and run the following command:

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

Restart the server

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

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

Run the server as a Microsoft Windows service

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

Register the service

About this task

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

Steps

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

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

Run multiple service instances

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

Deregister and uninstall

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

Log files

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

Uninstall the server

About this task

Use the `uninstall` command-line utility to uninstall the server using either interactive or non- interactive modes. Interactive mode provides options, progress, and a list of the files and directories that must be manually deleted if necessary.

Non-interactive mode, invoked with the `--no-prompt` option, suppresses progress information, except for fatal errors. All options for the `uninstall` command are listed with the `--help` option.

The `uninstall` command must be run as either the root user or the user account that installed the server.

Perform the following steps to uninstall in interactive mode:

Steps

1. Navigate to the server root directory.

```
$ cd PingData<server>
```

2. Start the uninstall command:

```
$ ./uninstall
```

3. Select the components to be removed, or press Enter to remove all components.
4. If the server is running, press Enter to shutdown the server before continuing.
5. Manually remove any remaining files or directories, if required.

Update servers in a topology

> ⓘ **Note:**
>
> This information is only applicable when updating from a server version that uses the admin backend to a server version that uses the topology registry. This is only the case when updating from a server version earlier than 7.x to a 7.x version, and is not applicable to any updates to server version 8.x or later.

> ⓘ **Tip:**
>
> For additional considerations, see the *Planning your upgrade guide*.

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

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

- Changes to clustered configurations are not allowed in mixed-version clusters. This applies to configuration in the `cn=Cluster,cn=config` subtree and only applies to servers with matching cluster names. Consider this when updating multiple servers in a cluster.
- To make clustered configuration changes in a mixed-version cluster, choose one of the following options:
  - Update each server to the same version.
  - Temporarily split up the cluster by changing the `cluster-name` property on the server instance configuration objects.
- After you have updated all the servers to the same version, you can again make clustered configuration changes and those changes will mirror across the topology.

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

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

```
  --set startTLSEnabled:true
```

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

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

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

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

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

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

Update the server

About this task

This procedure assumes that an existing version of the server is stored at `PingData-server-old`. Make sure a complete, readable backup of the existing system is available before upgrading the server. Also, make sure there is a clear backout plan and schedule.

Steps

1. Download the latest version of the server software and unzip the file. For this example, the new server is located in the `PingData-server-new` directory.
2. Use the `update` tool of the newly unzipped build to update the server. Make sure to specify the server instance that is being upgrading with the `--serverRoot` option. The server must be stopped for the update to be applied.

Reverting an update

If necessary, a server can be reverted to the previous version using the `revert-update` tool. The tool accesses a log of file actions taken by the `update` tool to put the file system back to its prior state. If multiple updates have been run, the `revert-update` tool can be used multiple times to revert to each

prior update sequentially. For example, the `revert-update` command can be run to revert to the server's previous state, then run again to return to its original state. The server is stopped during the revert-update process.

> ⓘ **Note:**
>
> Reverting an update is not supported for upgrades to version 7.0, due to the topology backend changes.

Use the `revert-update` tool in the server root directory revert back to the most recent version of the server:

```
$ PingData-server-old/revert-update
```

Revert an update

Once the server has been updated, you can revert to the most recent version (one level back) using the revert-update tool. The revert-update tool accesses a log of file actions taken by the updater to put the file system back to its prior state. If you have run multiple updates, you can run the revert-update tool multiple times to revert to each prior update sequentially. You can only revert back one level. For example, if you have run the update twice since first installing the server, you can run the revert-update command to revert to its previous state, then run the revert-update command again to return to its original state.

Revert from version 7.x to a version earlier than 7.0

Reverting from version 7.0 or later to a pre-7.0 version can be done using the `revert-update` command with some extra steps. This is also the case when updating or reverting from a pre-6.2.0.2 to 6.2.0.2 or later. These steps are listed when the `update` and `revert-update` tool are run as well. You may need to perform one or more of the following tasks, depending on your installation and configuration:

- When updating or reverting from 6.2.0.2 or later to a pre-6.2.0.2 version, indexes may need to be rebuilt. Older versions of the server use an incompatible format for Local DB Composite Indexes. To update a server with composite indexes in the previous format, delete these indexes and re-run the update. After the update is complete, recreate the indexes and use the `rebuild-index` tool to rebuild the indexes. The command for recreating an index will be in the "Undo" portion of the `logs/config-audit.log` file. If you wish to later revert to an older version, delete and recreate those composite indexes again after the revert has completed.
- When updating to 7.x for the first time, instance names will need to be set for each server in the topology if they were not previously set. This is done with the following `dsconfig` command:

  ```
  $ bin/dsconfig --bindDN "cn=Directory Manager" \
    --bindPassword secret \
    --no-prompt set-global-configuration-prop \
    --set instance-name:<name>
  ```

- Topology information such as server instances, instance and secret keys, server groups, and administrator users have moved to the topology portion of the configuration from the `admin` backend. As long as new servers are not added to the topology after this update, the `revert-update` command can be used to return to the previous version.

  However, if new servers are added, then the restored `admin` backend of this server will not contain information about the new servers, and the local server will not be able to communicate with any other servers in the topology. New servers should not be added to the topology if reverting this update is a possibility.
- If new servers were added to the topology after the update, the new servers must be temporarily removed from the topology until all servers have been reverted to the previous version.
- When a server is reverted to a pre-7.x version, any servers in the topology using the topology portion of the configuration (rather than the `admin` backend) will needto know that the reverted server was

downgraded to the `admin` backend. This is done by running the following `dsconfig` command on one of the servers that has not been reverted:

```
$ bin/dsconfig set-server-instance-prop \
  --instance-name <Reverted server instance name> \
  --set server-version:<Version to which server is reverted>
```

- If the topology does not have a master server when this command is run, it will not succeed. In this case, one of the remaining updated servers in the topology must be made master with the following command. This will enable the chosen instance to run the first command successfully.

```
$ bin/dsconfig set-global-configuration-prop \
  --set force-as-master-for-mirrored-data:true
```

- The 7.x server version includes database changes that are not compatible with previous server versions (6.x or older). If you wish to later revert to an older version, the data must be exported to LDIF before performing the reversion. Re-import the data after the revert process has completed. In addition, the `changelogDb/` and `db/changelog/` directories in the reverted server root must be deleted after the revert has completed.

When starting up the server for the first time after a revert has been run, and the necessary extra steps have been completed, the server will display warnings about "offline configuration changes," but they are not critical and will not appear on subsequent startups.

To revert to the most recent server version

Use `revert-update` in the server root directory revert back to the most recent version of the server.

```
$ <PingServer>-old/revert-update
```

Install a failover server

About this task

PingDataSync Server supports redundant failover servers that automatically become active when the primary server is not available. Multiple servers can be present in the topology in a configurable prioritized order.

Before installing a failover server, have a primary server already installed and configured. When installing the redundant server, the installer will copy the first server's configuration.

The primary and secondary server configuration remain identical. Both servers should be registered to the `allservers` group and all `dsconfig` changes need to be applied to the server group `allservers`.

ⓘ **Note:**

If the primary server has extensions defined, they should also be installed on any cloned or redundant servers. If extensions are missing from a secondary server, the following message is displayed during the installation:

```
Extension class <com.server.directory.sync.MissingSyncExtension> was not
found. Run manage-extension --install to install your extensions. Re-run
 setup to continue.
```

To remove a failover server, use the `uninstall` command.

Steps

1. Unpack the PingDataSync Server zip build. Name the unpacked directory something other than the first server instance directory.

```
$ unzip PingData<server>-<version>.zip -d <server2>
```

2. Navigate to the server root directory.
3. Use the `setup` tool in interactive mode in Install the Server, or in non-interactive mode as follows:

```
$ ./setup --localHostName <server2>.example.com --ldapPort 7389 \
  --masterHostName <server1>.example.com --masterPort 8389 \
  --masterUseNoSecurity \
  --acceptLicense \
  --rootUserPassword password \
  --no-prompt
```

The secondary server is now ready to take over as a primary server in the event of a failover. No `realtime-sync` invocations are needed for this server.

4. Verify the configuration by using the `bin/status` tool. Each server instance is given a priority index. The server with the lowest priority index number has the highest priority.

```
$ bin/status --bindPassword secret

...(status output)...
                                --- Sync Topology ---
Host:Port                          :Status :Priority
---------------------------------------:-------------:---------------
<server>.example.com:389 (this server) : Active      : 1
<server>.example.com:389         : Unavailable : 2
```

5. Obtain the name of a particular server, run the `dsconfig` tool with the `list-external-servers` option.

```
$ bin/dsconfig list-external-servers
```

6. To change the priority index of the server, use the `bin/dsconfig` tool:

```
$ bin/dsconfig set-external-server-prop \
  --server-name <server2>.example.com:389 \
  --set <server>-priority-index:1
```

Administrative accounts

Users that authenticate to the Configuration API or the Administrative Console are stored in `cn=RootDNs,cn=config`. The `setup` tool automatically creates one administrative account when performing an installation. Accounts can be added or changed with the `dsconfig` tool.

Change the administrative password

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

Steps

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

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

2. Use ldappasswordmodify to change the root user's password.

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

3. Remove the text file.

```
$ rm rootuser.txt
```

Configuring PingDataSync Server

PingDataSync Server provides a suite of tools to configure a single server instance or server groups. All configuration changes to the server are recorded in the `config-audit.log`. Before configuring PingDataSync Server, review *Sync Configuration Components.*

Topics include:

Configuration checklist

Prior to any deployment, determine the configuration parameters necessary for the Synchronization topology. Gather the following:

External servers

External Server Type

Determine the type of external servers included in the synchronization topology. See *Overview of PingDataSync Server* on page 1281 for a list of supported servers.

LDAP Connection Settings

Determine the host, port, bind DN, and bind password for each external server instance(s) included in the synchronization topology.

Security and Authentication Settings

Determine the secure connection types for each external server (SSL or StartTLS). Determine authentication methods for external servers such as simple, or external (SASL mechanisms). If synchronizing passwords, encoded or especially for clear-text, the connection should be secure. If synchronizing to or from a Microsoft Active Directory system, establish an SSL or StartTLS connection to PingDataSync Server. *Configuring password encryption* on page 1420 should also be enabled for synchronization from Active Directory, or when synchronizing clear-text passwords.

Sync pipes

A Sync Pipe defines a single synchronization path between the source and destination targets. One Sync Pipe is needed for each point-to-point synchronization path defined for a topology.

Sync Source

Determine which external server is the Sync Source for the synchronization topology. A prioritized list of external servers can be defined for failover purposes.

Sync Destination

Determine which external server is the Sync Destination for the synchronization topology. A prioritized list of external servers can be defined for failover purposes.

Sync classes

A Sync Class defines how attributes and DNs are mapped and how Source and Destination entries are correlated. For each Sync Pipe defined, define one or more Sync Classes with the following information:

Evaluation Order

If defining more than one Sync Class, the evaluation order of each Sync Class must be determined with the `evaluation-order-index` property. If there is an overlap between criteria used to identify a Sync Class, the Sync Class with the most specific criteria is used first.

Base DNs

Determine which base DNs contain entries needed in the Sync Class.

Include Filters

Determine the filters to be used to search for entries in the Sync Source.

Synchronized Entry Operations

Determine the types of operations that should be synchronized: creates, modifications, and/or deletes.

DNs

Determine the differences between the DNs from the Sync Source topology to the Sync Destination topology. Are there structural differences in each Directory Information Tree (DIT)? For example, does the Sync Source use a nested DIT and the Sync Destination use a flattened DIT?

Destination Correlation Attributes

Determine the correlation attributes that are used to associate a source entry to a destination entry during the synchronization process. During the configuration setup, one or more comma-separated lists of destination correlation attributes are defined and used to search for corresponding source entries. PingDataSync Server maps all attributes in a detected change from source to destination attributes using the attribute maps defined in the Sync Class.

The correlation attributes are flexible enough so that several destination searches with different combinations of attributes can be performed until an entry matches. For LDAP server endpoints, use the distinguished name (DN) to correlate entries. For example, specify the attribute lists `dn,uid`, `uid,employeeNumber` and `cn,employeeNumber` to correlate entries in LDAP deployments. PingDataSync Server will search for a corresponding entry that has the same `dn` and `uid` values. If the search fails, it then searches for `uid` and `employeeNumber`.

Again if the search fails, it searches for `cn` and `employeeNumber`. If none of these searches are successful, the synchronization change would be aborted and a message logged.

To prevent incorrect matches, the most restrictive attribute lists (those that will never match the wrong entry) should be first in the list, followed by less restrictive attribute lists, which will be used when the earlier lists fail. For LDAP-to-LDAP deployments, use the DN with a combination of other unique identifiers in the entry to guarantee correlation. For other non- LDAP deployments, determine the attributes that can be synchronized across the network.

Attributes

Determine the differences between the attributes from the Sync Source to the Sync Destination, including the following:

- Attribute Mappings – How are attributes mapped from the Sync Source to the Sync Destination? Are they mapped directly, mapped based on attribute values, or mapped based on attributes that store DN values?
- Automatically Mapped Source Attributes – Are there attributes that can be automatically synchronized with the same name at the Sync Source to Sync Destination? For example, can direct mappings be set for `cn`, `uid`, `telephoneNumber`, or for all attributes?
- Non-Auto Mapped Source Attributes – Are there attributes that should not be automatically mapped? For example, the Sync Source may have an attribute, `employee`, while the Sync Destination may have a corresponding attribute, `employeeNumber`. If an attribute is not automatically mapped, a map must be provided if it is to be synchronized.
- Conditional Value Mapping – Should some mappings only be applied some of the time as a function of the source attributes? Conditional value mappings can be used with the `conditional-value-pattern` property, which conditionalizes the attribute mapping based on the subtype of the entry, or on the value of the attribute. For example, this might apply if the `adminName` attribute on the destination should be a copy of the `name` attribute on the source, but only if the `isAdmin` attribute on the source is set to `true`. Conditional mappings are multi-valued. Each value is evaluated until one is matched (the condition is `true`). If none of the conditional mappings are matched, the ordinary mappings is used. If there is not an ordinary mapping, the attribute will not be created on the destination.

Sync user account

PingDataSync Server creates a Sync User account DN on each external server. The account (by default, `cn=Sync User`) is used exclusively by PingDataSync Server to communicate with external servers. The entry is important in that it contains the credentials (DN and password) used by PingDataSync Server to

access the source and target servers. The Sync User account resides in different entries depending on the targeted system:

- For the Ping Identity PingDirectory Server, Ping Identity PingDirectoryProxy Server, Nokia 8661 Directory Server, Nokia 8661 Directory Proxy Server, the Sync User account resides in the configuration entry (`cn=Sync User, cn=RootDNs,cn=config`).
- For Sun Directory Server, Sun DSEE, OpenDJ, Oracle Unified Directory, and generic LDAP directory topologies, the Sync User account resides under the base DN in the `userRoot` backend (`cn=SyncUser,dc=example,dc=com`).The Sync User account should not reside in the `cn=config` branch for Sun Directory Server and DSEE machines.
- For Microsoft Active Directory servers, the Sync User account resides in the Users container (`cn=Sync User,cn=Users,DC=adsync,DC=unboundid,DC=com`).
- For Oracle and Microsoft SQL Servers, the Sync User account is a login account (`SyncUser`)with the sufficient privileges to access the tables to be synchronized.

In most cases, modifications to this account are rare. Make sure that the entry is not synchronized by setting up an optional Sync Class if the account resides in the `userRoot` backend (Sun Directory Server or Sun DSEE) or Users container (Microsoft Active Directory). For example, a Sync Class can be configured to have all CREATE, MODIFY, and DELETE operations set to `false`.

Configure PingDataSync Server in Standard mode

The `create-sync-pipe-config` tool is used to configure Sync Pipes and Sync Classes. For bidirectional deployments, configure two Sync Pipes, one for each directional path.

Use the create-sync-pipe tool to configure synchronization on page 1310 illustrates a bidirectional synchronization deployment in standard mode. The example assumes that two replicated topologies are configured:

- The first endpoint topology consists of two Sun LDAP servers: the main server and one failover. Both servers have Retro change logs enabled and contain the full DIT that will be synchronized to the second endpoint.
- The second endpoint topology consists of two PingDirectory Servers: the main server and one failover. Both servers have change logs enabled and contain entries similar to the first endpoint servers, except that they use a `mail` attribute, instead of an `email` attribute.

A specific `mail` to `email` mapping must also be created to exclude the source attribute on the Sync Pipe going the other direction.

> ⓘ **Note:**
>
> If the source attribute is not excluded, PingDataSync Server will attempt to create an `email` attribute on the second endpoint, which could fail if the attribute is not present in the destination server's schema.

Then, two Sync Classes are defined:

- One to handle the customized `email` to `mail` attribute mapping.
- Another to handle all other cases (the default Sync Class).

The `dsconfig` command is used to create the specific attribute mappings. The `resync` command is used to test the mappings. Synchronization can start using the `realtime-sync` command.

Use the create-sync-pipe tool to configure synchronization

About this task

Use the `create-sync-pipe-config` utility to configure a Sync Pipe. Once the configuration is completed, settings can be adjusted using the `dsconfig` tool.

> ⓘ **Note:**
>
> If servers have no base entries or data, the `cn=Sync User,cn=Root DNs,cn=config` account needed to communicate cannot be created. Make sure that base entries are created on the destination servers.

If synchronizing pre-encoded passwords to a Ping PingDirectory Server destination, allow pre- encoded passwords in the default password policy. *Configuring password encryption* on page 1420 must also be configured on the destination. Be sure that the password encryption algorithm is supported by both source and destination servers with the following command:

```
$ bin/dsconfig set-password-policy-prop \
  --policy-name "Default Password Policy" \
  --set allow-pre-encoded-passwords:true
```

Encrypted and clear-text passwords can be synchronized by configuring the Sync Destination `password-synchronization-format`, and `require-secure-connection-for-clear-text-passwords` properties.

> ⓘ **Note:**
>
> The `require-secure-connection-for-clear-text-passwords` property can be set to `false` when working in a test environment. If the `password-synchronization-format` property is set to `clear-text`, and `require-secure-connection-for-clear-text-passwords` property is set to `true`, the connection must be secure. If a secure connection is not available, an error is generated and the password is not synchronized.

Perform the following steps to configure PingDataSync Server by using `create-sync-pipe-config`:

Steps

1. Start PingDataSync Server.

   ```
   $ <server-root>/bin/start-server
   ```

2. From the `bin` directory,run the `create-sync-pipe-config` tool.

   ```
   $bin/create-sync-pipe-config
   ```

3. On the Initial Synchronization Configuration Tool menu, press Enter (yes) to continue the configuration.
4. On the Synchronization Mode menu, press Enter to select Standard Mode.
5. On the Synchronization Directory menu, select **oneway(1)** or **bidirectional(2)** for the synchronization topology. This example assumes bidirectional synchronization.
6. On the Source Endpoint Type menu, select the directory or database server for the first endpoint.
7. On the Source Endpoint Name menu, type a name for the endpoint server, or press Enter to accept the default.
8. On the Base DNs menu, type the base DN on the first endpoint topology where the entries will be searched. In this example, (`dc=example,dc=com`) is used.
9. Select an option for the server security.
10. Type the host name and listener port number for the source server, or accept the default. Make sure that the endpoint servers are online and running.
11. Enter another server host and port, or press Enter to continue.
12. Enter the SyncUser account DN for the endpoint servers, or press Enter to accept the default (`cn=Sync User,cn=RootDNs,cn=config`).
13. Enter and confirm a password for this account.

14. The servers in the destination endpoint topology can now be configured. Repeat steps 6– 13 to configure the second server.
15. Define the maximum age of changelog log entries, or press Enter to accept the default.
16. After the source and destination topologies are configured, PingDataSync Server will "prepare" each external server by testing the connection to each server. This step determines if each account has the necessary privileges (root privileges are required) to communicate with and transfer data to each endpoint during synchronization.
17. Create a name for the Sync Pipe on the Sync Pipe Name menu, or press Enter to accept the default. Because this configuration is bidirectional, the following step is setting up a Sync Pipe path from the source endpoint to the destination endpoint. A later step will define another Sync Pipe from the PingDirectory Server to another server.
18. On the SyncClass Definitions menu, type `Yes` to create a custom Sync Class. A Sync Class defines the operation types (creates, modifies, or deletes), attributes that are synchronized, how attributes and DNs are mapped, and how source and destination entries are correlated.
19. Enter a name for the new Sync Class, such as "server1_to_server2."
20. On the Base DNs for Sync Class menu, enter one or more base DNs to synchronize specific subtrees of a DIT. Entries outside of the specified base DNs are excluded from synchronization. Make sure the base DNs do not overlap.
21. On the Filters for Sync Class menu, define one or more LDAP search filters to restrict specific entries for synchronization, or press Enter to accept the default (no). Entries that do not match the filters will be excluded from synchronization.
22. On the Synchronized Attributes for Sync Class menu, specify which attributes will be automatically mapped from one system to another. This example will exclude the source attribute (`email`) from being auto-mapped to the target servers.
23. On the Operations for Sync Class menu, select the operations that will be synchronized for the Sync Class, or press Enter to accept the default (`1,2,3`).
24. Define a default Sync Class that specifies how the other entries are processed, or press Enter to create a Sync Class called "Default Sync Class."
25. On the Default Sync Class Operations menu, specify the operations that the default Sync Class will handle during synchronization, or press Enter to accept the default.
26. Define a Sync Pipe going from the PingDirectory Server to the Sun Directory Server and exclude the `mail` attribute from being synchronized to the other endpoint servers.
27. Review the Sync Pipe Configuration Summary, and press Enter to accept the default (write configuration), which records the commands in a batch file (`<server- root>/sync-pipe-cfg.txt`). The batch file can be re-used to set up other topologies.

Next steps

Apply the configuration changes to the local PingDataSync Server instance by using a `dsconfig` batch file. Any Server SDK extensions, should be saved to the `<server-root>/lib/extensions` directory.

The next step will be to configure the attribute mappings using the `dsconfig` command.

Configuring attribute mapping

About this task

The following procedure defines an attribute map from the `email` attribute in the source servers to a `mail` attribute in the target servers. Both attributes must be valid in the target servers and must be present in their respective schemas.

ⓘ **Note:**

The following can also be done with `dsconfig` in interactive mode. Attribute mapping options are available from the PingDataSync Server main menu.

Steps

1. On PingDataSync Server, run the `dsconfig` command to create an attribute map for the "SunDS>DS" Sync Class for the "Sun DS to Ping Identity DS" Sync Pipe, and then run the second `dsconfig` command to apply the new attribute map to the Sync Pipe and Sync Class.

```
$ bin/dsconfig --no-prompt create-attribute-map \
  --map-name "SunDS>DS Attr Map" \
  --set "description:Attribute Map for SunDS>Ping Identity Sync Class" \
  --port 7389 \
  --bindDN "cn=admin,dc=example,dc=com" \
  --bindPassword secret
```

```
$ bin/dsconfig --no-prompt set-sync-class-prop \
  --pipe-name "Sun DS to DS" \
  --class-name "SunDS>DS" \
  --set "attribute-map:SunDS>DS Attr Map" \
  --port 7389 \
  --bindDN "cn=admin,dc=example,dc=com" \
  --bindPassword secret
```

2. Create an attribute mapping (from `email` to `mail`) for the new attribute map.

```
$ bin/dsconfig --no-prompt create-attribute-mapping \
  --map-name "SunDS>DS Attr Map" \
  --mapping-name mail --type direct \
  --set "description:Email>Mail Mapping" \
  --set from-attribute:email \
  --port 7389 \
  --bindDN "cn=admin,dc=example,dc=com" \
  --bindPassword secret
```

3. For a bidirectional deployment, repeat steps 1–2 to create an attribute map for the DS>SunDS Sync Class for the Ping Identity DS to Sun DS Sync Pipe, and create an attribute mapping that maps `mail` to `email`.

```
$ bin/dsconfig --no-prompt create-attribute-map \
  --map-name "DS>SunDS Attr Map" \
  --set "description:Attribute Map for DS>SunDS Sync Class" \
  --port 7389 \
  --bindDN "cn=admin,dc=example,dc=com" \
  --bindPassword secret
```

```
$ bin/dsconfig --no-prompt set-sync-class-prop \
  --pipe-name "Ping Identity DS to Sun DS" \
  --class-name "DS>SunDS" \
  --set "attribute-map:DS>SunDS Attr Map" \
  --port 7389 \
  --bindDN "cn=admin,dc=example,dc=com" \
  --bindPassword secret
```

```
$ bin/dsconfig --no-prompt create-attribute-mapping \
  --map-name "DS>SunDS Attr Map" \
  --mapping-name email \
  --type direct \
  --set "description:Mail>Email Mapping" \
```

```
  --set from-attribute:mail \
  --port 7389 \
  --bindDN "cn=admin,dc=example,dc=com" \
  --bindPassword secret
```

Configure server locations

About this task

PingDataSync Server supports endpoint failover, which is configurable using the `location` property on the external servers. By default, the server prefers to connect to, and failover to, endpoints in the same location as itself. If there are no location settings configured, PingDataSync Server will iterate through the configured list of external servers on the Sync Source and Sync Destination when failing over.

> ⓘ **Note:**
>
> Location-based failover is only applicable for LDAP endpoint servers.

Steps

1. On PingDataSync Server, run the `dsconfig` command to set the location for each external server in the Sync Source and Sync Destination. For example, the following command sets the location for six servers in two data centers, `austin` and `dallas`.

```
$ bin/dsconfig set-external-server-prop \
  --server-name example.com:1389 \
  --set location:austin
```

```
$ bin/dsconfig set-external-server-prop \
  --server-name example.com:2389 \
  --set location:austin
```

```
$ bin/dsconfig set-external-server-prop \
  --server-name example.com:3389 \
  --set location:austin
```

```
$ bin/dsconfig set-external-server-prop \
  --server-name example.com:4389 \
  --set location:dallas
```

```
$ bin/dsconfig set-external-server-prop \
  --server-name example.com:5389 \
  --set location:dallas
```

```
$ bin/dsconfig set-external-server-prop \
  --server-name example.com:6389 \
  --set location:dallas
```

2. Run `dsconfig` to set the location on the Global Configuration. This is the location of PingDataSync Server itself. In this example, set the location to "austin."

```
$ bin/dsconfig set-global-configuration-prop \
  --set location:austin
```

Use the Configuration API

PingData servers provide a Configuration API, which may be useful in situations where using LDAP to update the server configuration is not possible. The API is consistent with the System for Cross-domain Identity Management (SCIM) 2.0 protocol and uses JSON as a text exchange format, so all request headers should allow the `application/json` content type.

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

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

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

Authentication and authorization

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

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

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

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

Relationship between the Configuration API and the dsconfig tool

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

The API includes the filter, sort, and pagination query parameters described by the SCIM specification. Specific attributes may be requested using the `attributes` query parameter, whose value must be a comma-delimited list of properties to be returned, for example `attributes=baseDN,description`. Likewise, attributes may be excluded from responses by specifying the `excludedAttributes` parameter. See *Sorting and filtering configuration objects* on page 1324 for more information on query parameters.

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

HTTP Method	Description	Related dsconfig Example

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

POST	Creates a new instance of an object when used with a relation parent path, such as `config/backends`.	`create-backend`
PUT	Replaces the existing attributes of an object. A PUT operation is similar to a PATCH operation, except that the PATCH is determined by determining the difference between an existing target object and a supplied source object. Only those attributes in the source object are modified in the target object. The target object is specified using a path, such as `/config/backends/userRoot`.	`set-backend-prop` `set-global-configuration-prop`
PATCH	Updates the attributes of an existing object when used with a path representing an object, such as `/config/backends/userRoot`. See *PATCH Example* on page 1319.	`set-backend-prop` `set-global-configuration-prop`
DELETE	Deletes an existing object when used with a path representing an object, such as `/config/backends/userRoot`.	`delete-backend`

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

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

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

GET Example

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

```
GET /config/backends/userRoot
```

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

The response:


```
{
  "schemas": [
    "urn:unboundid:schemas:configuration:2.0:backend:local-db"
  ],
  "id": "userRoot",
  "meta": {
    "resourceType": "Local DB Backend",
    "location": "http://localhost:5033/config/backends/userRoot"
  },
  "backendID": "userRoot2",
  "backgroundPrime": "false",
  "backupFilePermissions": "700",
  "baseDN": [
    "dc=example2,dc=com"
  ],
  "checkpointOnCloseCount": "2",
  "cleanerThreadWaitTime": "120000",
  "compressEntries": "false",
  "continuePrimeAfterCacheFull": "false",
  "dbBackgroundSyncInterval": "1 s",
  "dbCachePercent": "10",
  "dbCacheSize": "0 b",
  "dbCheckpointerBytesInterval": "20 mb",
  "dbCheckpointerHighPriority": "false",
  "dbCheckpointerWakeupInterval": "1 m",
  "dbCleanOnExplicitGC": "false",
  "dbCleanerMinUtilization": "75",
  "dbCompactKeyPrefixes": "true",
  "dbDirectory": "db",
  "dbDirectoryPermissions": "700",
  "dbEvictorCriticalPercentage": "0",
  "dbEvictorLruOnly": "false",
  "dbEvictorNodesPerScan": "10",
  "dbFileCacheSize": "1000",
  "dbImportCachePercent": "60",
  "dbLogFileMax": "50 mb",
  "dbLoggingFileHandlerOn": "true",
  "dbLoggingLevel": "CONFIG",
  "dbNumCleanerThreads": "0",
  "dbNumLockTables": "0",
  "dbRunCleaner": "true",
  "dbTxnNoSync": "false",
  "dbTxnWriteNoSync": "true",
  "dbUseThreadLocalHandles": "true",
  "deadlockRetryLimit": "10",
  "defaultCacheMode": "cache-keys-and-values",
  "defaultTxnMaxLockTimeout": "10 s",
  "defaultTxnMinLockTimeout": "10 s",
  "enabled": "false",
  "explodedIndexEntryThreshold": "4000",
  "exportThreadCount": "0",
  "externalTxnDefaultBackendLockBehavior": "acquire-before-retries",
  "externalTxnDefaultMaxLockTimeout": "100 ms",
  "externalTxnDefaultMinLockTimeout": "100 ms",
  "externalTxnDefaultRetryAttempts": "2",
  "hashEntries": "false",
  "id2childrenIndexEntryLimit": "66",
  "importTempDirectory": "import-tmp",
```

```
  "importThreadCount": "16",
  "indexEntryLimit": "4000",
  "isPrivateBackend": "false",
  "javaClass": "com.unboundid.directory.server.backends.jeb.BackendImpl",
  "jeProperty": [
    "je.cleaner.adjustUtilization=false",
    "je.nodeMaxEntries=32"
  ],
  "numRecentChanges": "50000",
  "offlineProcessDatabaseOpenTimeout": "1 h",
  "primeAllIndexes": "true",
  "primeMethod": [
    "none"
  ],
  "primeThreadCount": "2",
  "primeTimeLimit": "0 ms",
  "processFiltersWithUndefinedAttributeTypes": "false",
  "returnUnavailableForUntrustedIndex": "true",
  "returnUnavailableWhenDisabled": "true",
  "setDegradedAlertForUntrustedIndex": "true",
  "setDegradedAlertWhenDisabled": "true",
  "subtreeDeleteBatchSize": "5000",
  "subtreeDeleteSizeLimit": "5000",
  "uncachedId2entryCacheMode": "cache-keys-only",
  "writabilityMode": "enabled"
}
```


GET list example

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

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

The response (which has been shortened):


```
{
  "schemas": [
    "urn:ietf:params:scim:api:messages:2.0:ListResponse"
  ],
  "totalResults": 24,
  "Resources": [
    {
      "schemas": [
        "urn:unboundid:schemas:configuration:2.0:backend:ldif"
      ],
      "id": "adminRoot",
      "meta": {
        "resourceType": "LDIF Backend",
        "location": "http://localhost:5033/config/backends/adminRoot"
      },
      "backendID": "adminRoot",
      "backupFilePermissions": "700",
      "baseDN": [
        "cn=admin data"
      ],
      "enabled": "true",
      "isPrivateBackend": "true",
      "javaClass": "com.unboundid.directory.server.backends.LDIFBackend",
      "ldifFile": "config/admin-backend.ldif",
      "returnUnavailableWhenDisabled": "true",
```

```
      "setDegradedAlertWhenDisabled": "false",
      "writabilityMode": "enabled"
    },
    {
      "schemas": [
        "urn:unboundid:schemas:configuration:2.0:backend:trust-store"
      ],
      "id": "ads-truststore",
      "meta": {
        "resourceType": "Trust Store Backend",
        "location": "http://localhost:5033/config/backends/ads-truststore"
      },
      "backendID": "ads-truststore",
      "backupFilePermissions": "700",
      "baseDN": [
        "cn=ads-truststore"
      ],
      "enabled": "true",
      "javaClass":
"com.unboundid.directory.server.backends.TrustStoreBackend",
      "returnUnavailableWhenDisabled": "true",
      "setDegradedAlertWhenDisabled": "true",
      "trustStoreFile": "config/server.keystore",
      "trustStorePin": "********",
      "trustStoreType": "JKS",
      "writabilityMode": "enabled"
    },
    {
      "schemas": [
        "urn:unboundid:schemas:configuration:2.0:backend:alarm"
      ],
      "id": "alarms",
      "meta": {
        "resourceType": "Alarm Backend",
        "location": "http://localhost:5033/config/backends/alarms"
      },
  ...
```

PATCH Example

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

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

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

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

  is analogous to:

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

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

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

  is analogous to:

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

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

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

  is analogous to:

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

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

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

  is analogous to:

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

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

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

  is analogous to:

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

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

Example request:

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

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

Example response:

```
{
  "schemas": [
    "urn:unboundid:schemas:configuration:2.0:backend:local-db"
  ],
  "id": "userRoot2",
  "meta": {
    "resourceType": "Local DB Backend",
    "location": "http://example.com:5033/config/backends/userRoot2"
  },
  "backendID": "userRoot2",
  "backgroundPrime": "false",
  "backupFilePermissions": "700",
  "baseDN": [
    "dc=example2,dc=com"
  ],
  "checkpointOnCloseCount": "2",
  "cleanerThreadWaitTime": "120000",
  "compressEntries": "false",
  "continuePrimeAfterCacheFull": "false",
  "dbBackgroundSyncInterval": "1 s",
  "dbCachePercent": "10",
  "dbCacheSize": "0 b",
  "dbCheckpointerBytesInterval": "20 mb",
  "dbCheckpointerHighPriority": "false",
  "dbCheckpointerWakeupInterval": "1 m",
  "dbCleanOnExplicitGC": "false",
  "dbCleanerMinUtilization": "75",
  "dbCompactKeyPrefixes": "true",
  "dbDirectory": "db",
  "dbDirectoryPermissions": "700",
  "dbEvictorCriticalPercentage": "0",
  "dbEvictorLruOnly": "false",
  "dbEvictorNodesPerScan": "10",
  "dbFileCacheSize": "1000",
  "dbImportCachePercent": "60",
  "dbLogFileMax": "50 mb",
  "dbLoggingFileHandlerOn": "true",
  "dbLoggingLevel": "CONFIG",
  "dbNumCleanerThreads": "0",
```

```
  "dbNumLockTables": "0",
  "dbRunCleaner": "true",
  "dbTxnNoSync": "false",
  "dbTxnWriteNoSync": "true",
  "dbUseThreadLocalHandles": "true",
  "deadlockRetryLimit": "10",
  "defaultCacheMode": "cache-keys-and-values",
  "defaultTxnMaxLockTimeout": "10 s",
  "defaultTxnMinLockTimeout": "10 s",
  "description": "123",
  "enabled": "false",
  "explodedIndexEntryThreshold": "4000",
  "exportThreadCount": "0",
  "externalTxnDefaultBackendLockBehavior": "acquire-before-retries",
  "externalTxnDefaultMaxLockTimeout": "100 ms",
  "externalTxnDefaultMinLockTimeout": "100 ms",
  "externalTxnDefaultRetryAttempts": "2",
  "hashEntries": "false",
  "importTempDirectory": "import-tmp",
  "importThreadCount": "16",
  "indexEntryLimit": "4000",
  "isPrivateBackend": "false",
  "javaClass": "com.unboundid.directory.server.backends.jeb.BackendImpl",
  "jeProperty": [
  "\"je.env.backgroundReadLimit=0\""
  ],
  "numRecentChanges": "50000",
  "offlineProcessDatabaseOpenTimeout": "1 h",
  "primeAllIndexes": "true",
  "primeMethod": [
    "none"
  ],
  "primeThreadCount": "2",
  "primeTimeLimit": "0 ms",
  "processFiltersWithUndefinedAttributeTypes": "false",
  "returnUnavailableForUntrustedIndex": "true",
  "returnUnavailableWhenDisabled": "true",
  "setDegradedAlertForUntrustedIndex": "true",
  "setDegradedAlertWhenDisabled": "true",
  "subtreeDeleteBatchSize": "5000",
  "subtreeDeleteSizeLimit": "5000",
  "uncachedId2entryCacheMode": "cache-keys-only",
  "writabilityMode": "enabled",
  "urn:unboundid:schemas:configuration:messages:2.0": {
    "requiredActions": [
      {
        "property": "jeProperty",
        "type": "componentRestart",
        "synopsis": "In order for this modification to take effect,
         the component must be restarted, either by disabling and
         re-enabling it, or by restarting the server"
      },
      {
        "property": "id2childrenIndexEntryLimit",
        "type": "other",
        "synopsis": "If this limit is increased, then the contents
         of the backend must be exported to LDIF and re-imported to
         allow the new limit to be used for any id2children keys
         that had already hit the previous limit."
      }
    ]
  }
}
```

API paths

The Configuration API is available under the `/config` path. A full listing of root sub-paths can be obtained from the `/config/ResourceTypes` endpoint:

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

Sample response (abbreviated):

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

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

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

Sorting and filtering configuration objects

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

GET Parameters for Sorting and Filtering

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

Update properties

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

Request:

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

Response:

```
{
  "schemas": [
    "urn:unboundid:schemas:configuration:2.0:backend:local-db"
  ],
  "id": "userRoot",
  "meta": {
    "resourceType": "Local DB Backend",
    "location": "http://example.com:5033/config/backends/userRoot"
  },
  "backendID": "userRoot",
  "backgroundPrime": "false",
  "backupFilePermissions": "700",
```

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

```
  "subtreeDeleteSizeLimit": "100000",
  "uncachedId2entryCacheMode": "cache-keys-only",
  "writabilityMode": "enabled"
}
```

Administrative actions

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

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

```
"urn:unboundid:schemas:configuration:messages:2.0": {
  "requiredActions": [
    {
      "property": "baseContextPath",
      "type": "componentRestart",
      "synopsis": "In order for this modification to
         take effect, the component must be restarted,
         either by disabling and re-enabling it, or by
         restarting the server"
    },
    {
      "property": "id2childrenIndexEntryLimit",
      "type": "other",
      "synopsis": "If this limit is increased, then the
         contents of the backend must be exported to LDIF
         and re-imported to allow the new limit to be used
         for any id2children keys that had already hit the
         previous limit."
    }
  ]
}
...
```

Update servers and server groups

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

```
https://example.com:5033/config/Backends/userRoot?applyChangeTo=singleServer
```

> ⓘ **Note:**
>
> This does not apply to mirrored subtree objects, which include Topology and Cluster level objects. Changes made to mirrored objects are applied to all objects in the subtree.

Configuration API responses

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

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

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

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

Configuration with the dsconfig tool

The Ping Identity servers provide several command-line tools for management and administration. The command-line tools are available in the `bin` directory for UNIX or Linux systems and `bat` directory for Microsoft Windows systems.

The `dsconfig` tool is the text-based management tool used to configure the underlying server configuration. The tool has three operational modes:

- Interactive Mode
- Non-interactive mode
- Batch mode

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

Each command-line utility provides a description of the subcommands, arguments, and usage examples needed to run the tool. View detailed argument options and examples by typing `-- help` with the command.

```
$ bin/dsconfig --help
```

To list the subcommands for each command:

```
$ bin/dsconfig --help-subcommands
```

To list more detailed subcommand information:

```
$ bin/dsconfig list-log-publishers --help
```

Use dsconfig in interactive mode

Running `dsconfig` in interactive command-line mode provides a menu-driven interface for accessing and configuring the PingData server. To start `dsconfig` in interactive mode, run the tool without any arguments:

```
$ bin/dsconfig
```

Running the tool requires server connection and authentication information. After connection information is confirmed, a menu of the available operation types is displayed.

Use dsconfig in non-interactive mode

Non-interactive command-line mode provides a simple way to make arbitrary changes to the server, and to use administrative scripts to automate configuration changes. To make changes to multiple configuration objects at the same time, use batch mode.

The general format for the non-interactive command line is:

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

The `--no-prompt` argument specifies non-interactive mode. The `{globalArgs}` argument provides a set of arguments that specify how to connect and authenticate to the server. Global arguments can be standard LDAP connection parameters or SASL connection parameters depending on the implementation. The `{subcommand}` specifies which general action to perform. The following uses standard LDAP connections:

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

The following uses SASL GSSAPI (Kerberos) parameters:

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

The `{subcommandArgs}` argument contains a set of arguments specific to the particular task. To always display the advanced properties, use the `--advanced` command-line option.

> ⓘ **Note:**
>
> Global arguments can appear anywhere on the command line. The subcommand-specific arguments can appear anywhere after the subcommand.

Use dsconfig batch mode

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

```
$ bin/dsconfig --no-prompt \
  --hostname host1 \
  --port 1389 \
  --bindDN "uid=admin,dc=example,dc=com" \
  --bindPassword secret \
  --batch-file /path/to/sync-pipe-config.txt
```

If a `dsconfig` command has a missing or incorrect argument, the command will fail and stop the batch process without applying any changes to the server. A `--batch-continue-on-error`option is available, which instructs `dsconfig` to apply all changes and skip any errors.

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

PingDataSync Server also provides a `docs/sun-ds-compatibility.dsconfig` file for migrations from Sun/Oracle to Ping Identity PingDataSync Server machines.

Topology configuration

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

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

Topology master requirements and selection

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

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

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

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

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

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

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

Topology components

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

Server configuration settings

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

The Global Configuration object contains a single topology setting, `force-as-master-for-mirrored-data`. This should be set to `true` on only one of the servers in the topology, and is used only if a situation occurs where the topology cannot determine a master because a majority of servers is not available. A server with this setting enabled will be assigned the role of master, if no suitable master can be determined. See *Topology master requirements and selection* on page 1330 for details about how a master is selected for a topology.

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

- `mirrored-subtree-peer-polling-interval` – Specifies the frequency at which the server polls its topology peers to determine if there are any changes that may warrant a new master selection.

A lower value will ensure a faster failover, but it will also cause more traffic among the peers. The default value is five seconds. If no suitable master is found, the polling frequency is adjusted to 100 milliseconds until a new master is selected.

- `mirrored-subtree-entry-update-timeout` – Specifies the maximum length of time to wait for an update operation (add, delete, modify or modify-dn) on an entry to be applied by the master on all of the servers in the topology. The default is 10 seconds. In reality, updates can take up to twice as much time as this timeout value if master selection is in progress at the time the update operation was received.
- `mirrored-subtree-search-timeout` – Specifies the maximum length of time in milliseconds to wait for search operations to complete. The default is 10 seconds.

Topology settings

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

Monitor data for the topology

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

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

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

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

Servers and certificates

The following provides a detailed description of PingDirectory Server certificate types.

PingDirectory Server uses two main types of certificates:

Listener certificates

Used to secure communication through TLS.

Inter-server certificates

Used to authenticate between server instances in a topology.

Listener certificates
When a client initiates TLS negotiation with the server, the server presents a certificate chain to the client and the certificate at the head of the chain functions as a listener certificate.

Because the client decides whether to trust the certificate chain, it is recommended that the chain be signed by an issuer whom the client is likely to trust or that the client can be easily configured to trust.

You can create self-signed certificates with long lifespans, but a certificate that a certification authority signs is likely to have a relatively short lifespan. Commercial authorities typically issue certificates that are valid for only one or two years, but some authorities use shorter validity windows.

Short certificate lifespans offer some security benefits. In particular, because most clients do not verify whether a certificate has been revoked, a shorter validity window minimizes the timeframe that a compromised certificate can be used. If the process for replacing certificates is streamlined or automated, administrative inconvenience can be kept to a minimum.

Listener certificates are stored in key stores that are referenced by key manager providers, which in turn provide the logic and configuration for accessing the key stores. If a server component, like a connection handler, requires access to a certificate that it presents to a peer during the TLS negotiation process, that component must reference the key manager provider that points to the key store containing the appropriate certificate. If the key store contains multiple certificates, and if the component referencing the key store includes a property specifying the certificate's nickname, the certificate with that alias is selected. Otherwise, the server lets the Java virtual machine (JVM) select a certificate that might not be well-defined.

The server also provides trust manager providers, which determine whether to trust the certificate chains with which it is presented. A trust manager provider can reference a specified trust store file, but other options include the JVM default trust store, which uses the Java installation's default set of trusted issuers, and the blind trust manager provider, which automatically trusts every certificate chain that is presented to it.

> ⓘ **Note:**
>
> Never use a blind trust manager in a production environment because it leaves the server vulnerable to impersonation and man-in-the-middle attacks. However, a blind trust manager can be convenient in test environments when troubleshooting certain types of problems.

Replacing listener certificates
Certificate authorities typically restrict the lifespans of the certificates that they sign. If you use a certification authority to issue listener certificates, you are likely replacing the certificates on a regular basis.

About this task

The **`replace-certificate`** tool performs the following steps:

1. Obtain a new certificate chain.
2. Make necessary updates to the key manager provider and the connection handler configurations
3. Update the server instance listener configuration with the new certificate.

The **`replace-certificate`** tool offers the following modes of operation:

Interactive mode

Walks you through the process of obtaining a new certificate and installing it in the server. Interactive mode also displays the non-interactive commands that are required to achieve the same result.

Non-interactive mode

Useful when scripting the process of replacing a certificate.

Steps

- To replace a listener certificate, run the **`replace-listener-certificate`** subcommand of the **`replace-certificate`** tool.

> ⓘ **Note:**
>
> You can replace certificates manually, but the **`replace-certificate`** tool automates the process. The **`replace-certificate`** tool provides information about multiple listener certificates during the transitional phase that occurs when you install them.
>
> The **`replace-listener-certificate`** subcommand takes arguments that provide the following information:
>
> - Arguments required to authenticate to PingDataSync Server, such as --bindDN and --bindPasswordFile
> - Details about the key store that contains the new certificate
> - Updates that must be made to the key and trust manager providers
> - Whether to signal the HTTP connection handler to reload its certificates after the update is complete

The following arguments are available:

Argument	Description
--source-key-store-file {path}	Path to the Java KeyStore (JKS) or PKCS #12 file that contains the private key entry with the new certificate chain. This argument is required.
--source-key-store-password {password}	Clear-text password that is needed to access the contents of the source key store.
--source-key-store-password-file {path}	Path to the file that contains the password necessary to access the contents of the source key store. The file can contain the password in the clear or can be encrypted with a definition from the server's encryption settings database.
--source-certificate-alias {alias}	Password that is required to access the appropriate private key in the source key store. If neither the --source-private-key-password nor the --source-private-key-password-file argument is provided, the key store password is used as the private key password.
--source-private-key-password-file {path}	Path to the file that contains the password needed to access the appropriate private key in the source key store. The file can contain the password in the clear or can be encrypted with a definition from the server's encryption settings database. If neither the --source-private-key-password nor the --source-private-key-password-file argument is provided, the key store password is used as the private key password.
--key-manager-provider {name}	Name of the key manager provider that is updated to use the new certificate chain. The value must identify a file-based key manager provider, and the new certificate chain must be enabled. Defaults to JKS if a value is not specified.

Argument	Description
--trust-manager-provider {name}	Name of the trust manager provider that is updated with the information required to trust the new certificate chain. The value must identify a file-based trust manager provider, and the new certificate chain must be enabled. If neither this argument nor the --use-jvm-default-trust-manager-provider argument is provided, the tool assumes that the name of the trust manager provider is identical to the name of the key manager provider.
--use-jvm-default-trust-manager-provider	Indicates that the server must be configured to use the JVM-default trust manager provider, which trusts certificates signed by issuers in the `cacerts` trust store provided with the Java virtual machine (JVM), rather than updating an existing trust manager provider.
--target-certificate-alias {alias}	Alias to use for the new certificate in the key manager provider's key store, and for appropriate updates in the trust manager provider's trust store. Defaults to an alias of server-cert if a value is not specified. ⓘ **Note:** If the key manager provider's key store, or the trust manager provider's trust store, already contains an entry with the given alias, the existing entry is renamed.
--reload-http-connection-handler-certificates	Indicates that the tool is requesting that the server cause any HTTPS-based connection handlers to reload their certificates, so that the connection handlers can use the updated certificate. LDAP connection handlers react to the change immediately and start presenting the new certificate chain during subsequent TLS negotiations. HTTPS connection handlers continue using the former certificate until the connection handler is restarted or until the connection handler is asked specifically to reload its certificates. ⓘ **Note:** This option might prevent clients with existing TLS sessions that were negotiated with the former certificate from being resumed.

- To remove earlier certificates from the server instance listener configuration, run the purge-retired-listener-certificates subcommand.

> ⓘ **Note:**
>
> The purge-retired-listener-certificates subcommand does not take arguments other than the ones that are required to authenticate to the server.
>
> By default, the **`replace-certificate`** tool updates the server instance listener configuration object to include the new listener certificate, and it merges the old and new certificates residing in the configuration object.

Inter-server certificates
Each server instance in a topology has an inter-server certificate that is generated during the setup process.

The inter-server certificate is not exposed to clients, so a trusted issuer does not need to sign it. Instead, the topology registry, representing a mirrored portion of the configuration with information about every PingDirectory Server instance in the environment, contains the information that each instance needs to trust the inter-server certificates for all other instances.

Inter-server certificates can be used to protect certain secrets that are shared among servers within the topology, like the secrets that are used to digitally sign log files, backups, and LDIF exports. Inter-server certificates include the encryption keys that reversible password-storage schemes use.

The inter-server certificate is generated with a long lifespan. Replace it only when you suspect that its private key is compromised.

Replacing the inter-server certificate
During the installation process, the inter-server certificate is generated with a long lifespan and does not require replacement under normal circumstances. You should replace the inter-server certificate only if you suspect that its private key is compromised.

About this task

The inter-server certificate is intended for use only between server instances within the same topology. Because it is not exposed to regular clients, the inter-server certificate does not need to be trusted.

The **`replace-certificate replace-inter-server-certificate`** command performs the following steps:

- Acquires the new inter-server certificate from a provided Java KeyStore (JKS) or PKCS #12 key store
- Makes the necessary updates to the `config/ads-truststore` file in the server key store
- Updates the server instance configuration object to include the new inter-server certificate

> ⓘ **Note:**
>
> To avoid the need to replace the inter-server certificate on a regular basis,use a self-signed certificate with a long lifespan. Each server instance must possess its own, unique inter-server certificate that satisfies the following conditions:
>
> - Uses an RSA key pair
> - Has a minimum key size of 2048 bits
>
> The following types of certificates are not allowed:
>
> - Certificates with an elliptic curve key pair
> - Certificates with an RSA key that is smaller than 2048 bits

Steps

- To replace the inter-server certificate, run the **`replace-inter-server-certificate`** subcommand of the **`replace-certificate`**.

  The **`replace-inter-server-certificate`** subcommand takes a subset of the arguments that are used with the **`replace-listener-certificate`** subcommand, including the following arguments:

  - --source-key-store-file {path}--source-key-store-password {password}
  - --source-key-store-password-file {path}
  - --source-certificate-alias {alias}
  - --source-private-key-password {password}
  - --source-private-key-password-file {path}
- To delete earlier values that are no longer needed, run the **`purge-retired-inter-server-certificates`** subcommand.

> ⓘ **Note:**
>
> By default, the new inter-server certificate is merged with the existing values in the server instance configuration entry.

X.509 certificates
PingDataSync Server supports X.509 certificates, the most common type of certificates. *RFC 5280* describes X.509v3, which provides the current version of the specification.

An X.509v3 certificate includes the following components:

X.509 encoding version

Enables the differentiation between an X.509v3 certificate and one that conforms to an earlier or later version of the specification.

Serial number of the certificate

Integer value that uniquely identifies a certificate as issued by a certification authority.

Subject DN

Distinguished name for the certificate, which often provides details about the context in which the certificate is to be used. For more information, see *Certificate subject DNs* on page 319.

Issuer DN

Distinguished name for the issuer certificate, which is the certificate used to sign the certificate. For a self-signed certificate, this value matches the subject DN.

Validity window

Indicates the timeframe during which the certificate is considered valid. This component includes the following elements:

- notBefore

  Specifies the earliest time at which the certificate is considered valid.
- notAfter

  Specifies the latest time at which the certificate is considered valid.

Public key

Public portion of a pair of cryptographically linked keys. For more information, see *Certificate key pairs* on page 320.

Signature

A type of cryptographic proof that the certificate truly was sent from the issuer and has remained unaltered. A self-signed certificate is signed with its own private key. Otherwise, it is signed with the issuer's private key.

An X.509v3 certificate might also include the following optional components:

Subject unique ID

Uniquely identifies the certificate. This component has been deprecated in favor of the subject key identifier extension, so it is generally omitted from X.509v3 certificates.

Issuer unique ID

Subject unique ID of the issuer certificate, if available. This component has been deprecated in favor of the authority key identifier extension.

Set of extensions

Provides additional context for the certificate and the manner in which it is used. For more information, see *Certificate extensions* on page 320.

Certificate subject DNs

A certificate's subject distinguished name (DN) provides information about how the certificate should be used.

Like an LDAP DN, a certificate's subject DN consists of a comma-delimited series of attribute-value pairs. However, unlike an LDAP DN, the attribute names in a certificate subject DN are typically written in all uppercase characters.

A certificate's subject DN is also referred to as its subject. The following attributes commonly appear in a certificate subject.

Attribute name	Attribute description
`CN`	Common name
	Note: For a listener certificate, the `CN` attribute typically identifies the host name that clients use to access the certificate. However, the subject alternative name extension is recommended more highly for accomplishing the same task. Most certificate subject DNs include at least the `CN` attribute.
`E`	Email address
`OU`	Name of the organizational unit, such as the relevant department
`O`	Name of the organization or company
`L`	Name of the locality, such as the appropriate city
`ST`	Full name of the state or province
`C`	ISO 3166 country code

A certificate subject includes at least one attribute-value pair, and the `CN` attribute is typically present. Other attributes can be omitted, although the `O` and `C` attributes are also common. For example, a listener certificate for a server with an address of `ldap.example.com`, which is run by the US-based company Example Corp, might have a subject of `CN=ldap.example.com,O=Example Corp,C=US`.

Certificate key pairs
Each certificate contains a key pair that consists of two keys that are linked cryptographically. If you encrypt data with one key, the data can be only decrypted with the other key.

Although a key pair can be created easily when both keys are generated simultaneously, the process of deriving one key from the other is extremely difficult, a process categorized in cryptographic terms as computationally infeasible.

When generating a key pair, one key is designated as the public key, and the other key is designated the private key. The public key can be made widely available, but the private key must be kept secret and not shared with anyone.

As long as the secrecy of the private key is maintained, the key pair can be used to perform the following functions:

- Encryption, sometimes referred to as confidentiality

  If someone wants to send you a secret message without anyone else viewing it, the message can be encrypted with your public key. Only you possess the private key, so only you can decrypt the message.
- Digital signatures

  If you encrypt data with your private key, it can be decrypted only with your public key. Because your public key can be made widely available, this encryption method does not actually protect the content. However, digital signatures prove that a message came from you because only your private key could have generated it.

> ⓘ **Note:**
>
> When generating a digital signature, the entire message is generally not encrypted. Only a hash of the message is encrypted, typically by using a digest algorithm like SHA-256.
>
> This approach protects the integrity of a message. A decrypted signature that matches the digest of the original message guarantees that the message came from you and that it has remained unaltered since you signed it.

The following public key algorithms are used primarily in certificates that facilitate TLS communication:

- RSA, which is based on the multiplication of large prime numbers
- EC, which is based on computations that involve special types of elliptical curves

Although RSA is supported more widely than EC, it is slower and requires larger keys to achieve the same level of security. To support legacy clients, you should use an RSA certificate and choose a key size of at least 2,048 bits.

If all of your clients support EC certificates, you should use an EC certificate with a key size of at least 256 bits.

Certificate extensions
Extensions provide additional context for a certificate.

Some of the more common extension types include the following:

Subject key identifier

Holds a unique identifier for the certificate, which is generally derived from the certificate's public key.

Authority key identifier

Holds the subject key identifier for the issuer certificate. This extension type helps to identify the issuer certificate, especially when presented with an incomplete certificate chain.

Subject alternative name

Holds a list of ways that clients are expected to reference a server when establishing a connection to it.

> ⓘ **Note:**
>
> Clients must take this information into account when deciding whether to trust a server's certificate.
>
> The most common types of values include DNS names, IP addresses, and URIs. DNS names must be fully qualified, but can optionally use an asterisk in the leftmost component to match any single name in that component. For example, `*.example.com` could match `www.example.com` or `ldap.example.com`, but would not match `ldap.east.example.com` or `example.com`.

Key usage

Provides information about the manner in which the certificate is expected to be used. The following key usages are allowed:

`digitalSignature`

Indicates that the certificate can be used for digitally signing data, excluding certificates and certificate revocation lists (CRL).

`nonRepudiation`

Indicates that the certificate can be used to prevent denying the authenticity of a message. `nonRepudiation` is also known as `contentCommitment`.

`keyEncipherment`

Indicates that the certificate can be used to protect encryption keys, such as symmetric keys that are derived during TLS key agreement.

`dataEncipherment`

Indicates that the certificate can be used for encrypting data directly.

`keyAgreement`

Indicates that the certificate's public key can be used for key agreement, such as deriving the symmetric key that protects TLS communication.

`keyCertSign`

Indicates that the certificate can act as a certification authority and be used for signing other certificates.

`cRLSign`

Indicates that the certificate can be used to sign CRLs.

`encipherOnly`

When used in conjunction with `keyEncipherment`, indicates that the public key can be used only for encrypting data during key agreement.

`decipherOnly`

When used in conjunction with `keyEncipherment`, indicates that the public key can be used only for decrypting data during key agreement.

Extended key usage

Acts as an alternative to the key usage extension and provides additional high-level functionality. The following extended key usages are allowed:

`serverAuth`

Indicates that the server can present the certificate to the client during TLS negotiation.

`clientAuth`

Indicates that the client can present the certificate to the server during TLS negotiation.

`codeSigning`

Indicates that the certificate can be used to sign source and compiled code.

`emailProtection`

Indicates that the certificate can be used to sign or encrypt email messages.

`timeStamping`

Indicates that the certificate can be used to assert the time that an event occurred.

`ocspSigning`

Indicates that the certificate can be used to sign an online certificate status protocol (OCSP) response.

Basic constraints

Indicates whether the certificate can act as a certification authority and, if so, the maximum number of intermediate certificates that can follow it in a certificate chain.

Certificate chains

A certificate chain is an ordered list of one or more certificates. In such a chain, each subsequent certificate is the issuer of the previous certificate.

During TLS negotiation, the server presents a certificate chain to the client, which determines whether to trust the chain and continue with the negotiation. The client can also present its own certificate chain to the server.

If a certificate is self-signed, its chain contains only that single certificate. If a certificate is signed by a self-signed certificate authority (CA) certificate, such as a root CA, the chain contains two certificates: the server certificate and the CA certificate that follows it. If a single intermediate CA (a CA certificate that is signed by a root CA) is present, the chain contains the server certificate, followed by the intermediate CA, and then the root CA.

Intermediate certificate authorities are useful for security purposes, especially in commercial authorities. If a client trusts a root CA certificate, it is likely to trust anything with that root CA certificate at the base of its chain. Consequently, the root CA certificate must be kept secure.

ⓘ **Note:**

If the root CA certificate is compromised, any certificate that is directly or indirectly signed by it can no longer be trusted.

With intermediate CA certificates, the root certificate can be kept offline in secure storage and used only when a new intermediate CA certificate must be signed. The intermediate CA certificates can be used to sign end-entity certificates, but must be protected to avoid compromising any of the certificates. A compromised certificate must be revoked along with all of the certificates that it signed. In such a scenario, the root CA can be used to sign a new certificate.

ⓘ **Note:**

The certificate chain that the server presents to the client, or that the client presents to the server, during TLS negotiation does not always need to be the complete chain. If the root CA at the end of the chain is widely trusted, the server can assume that the client already has that root CA in its default set of trusted certificates. The server can leave that root CA off the chain with the assumption that the client will retrieve

it from its default trust store. While the same assumption could theoretically be true for intermediate CA certificates, only the root CA certificate is commonly omitted. When a client receives an incomplete chain, the client looks in its default trust store to determine whether the trust store contains the issuer certificate, which it can identify by using properties like the issuer distinguished name (DN) or an authority key identifier extension.

The certificate at the head of a certificate chain, which appears as the first one in the list, is often called the end-entity certificate. If this certificate appears at the head of the chain that a server presents during TLS negotiation, it is referred to as the server certificate. If the certificate appears at the head of a chain that a client presents, it is referred to as a client certificate. The certificate at the end of a complete chain must be a root CA certificate. In the case of a self-signed certificate, the chain contains only a single certificate that serves both roles.

About representing certificates, private keys, and certificate signing requests

X.509 is an encoding format that uses the ASN.1 distinguished encoding rules (DER), which exist in binary format. When writing a certificate to a file, either a raw DER format or a plaintext format called PEM can be used.

PEM encoding consists of a line that contains the text `-----BEGIN CERTIFICATE-----`, followed by a set of lines that contains the base64-encoded representation of the raw DER bytes (typically with no more than 64 characters per line), followed by a line that contains the text `-----END CERTIFICATE-----`.

The X.509 encoding contains a certificate's public key, but not its private key. The PKCS #8 specification in *RFC 5958* describes the encoding for private keys. This approach uses a DER encoding with a PEM variant that instead uses the following header and footer, respectively.

```
 -----BEGIN PRIVATE KEY-----
-----END PRIVATE KEY-----
```

RFC 5958 also describes an encrypted representation of the private key, but that format is currently unsupported.

The PKCS #10 specification in *RFC 2986* describes the CSR format. This format uses a DER encoding with a PEM variant that uses the following header and footer, respectively.

```
-----BEGIN CERTIFICATE REQUEST-----
-----END CERTIFICATE REQUEST-----
```

Some implementations use the following alternate, nonstandard forms.

```
-----BEGIN NEW CERTIFICATE REQUEST-----
-----END NEW CERTIFICATE REQUEST-----
```

Certificate trust

When a server presents its certificate chain to a client during TLS negotiation, the client decides whether to trust the certificate chain and concludes whether it is communicating with a legitimate server instead of an impostor.

If a client is tricked through DNS hijacking into communicating with a rogue application instead of with a legitimate server, the application can steal the client's credentials, or can fool the client into concluding that it has performed an action that it has not performed. If a rogue application acts as a broker between the client and the legitimate server, the client might be unable to detect the change, and the malicious application might be capable of stealing data or altering the communication. Consequently, you should avoid `trust all` or `blind trust` options in a production environment.

When determining whether to trust a server certificate chain, a client performs the following steps.

Processing steps

1. Verifies that it has received the complete certificate chain.

   If a server presents an incomplete chain, the client must ensure that it can complete the chain with information in an explicitly provided trust store or default trust store. If the client cannot complete the certificate chain, the chain is not trusted.
2. Verifies that each subsequent certificate in the chain is the issuer certificate for, and that its private key was used to sign, the certificate that precedes it.

   > ⓘ **Note:**
   >
   > If a certificate chain contains extraneous certificates, or if a subsequent certificate did not issue the certificate that precedes it, the chain is not trusted.
3. Confirms that it has a reason to trust the certificate at the root of the chain.

   > ⓘ **Note:**
   >
   > This step is generally performed by ensuring that the root certificate authority (CA) certificate can be found in either a default trust store or a trust store that is configured for use by the client. If the client has no prior knowledge of the root CA certificate, the chain is not trusted.
4. Verifies that the current time lies within the validity window for each certificate in the chain.

   > ⓘ **Note:**
   >
   > The chain is not trusted under the following conditions:
   >
   > - When the `notBefore` value of any certificate in the chain is later than the current time.
   > - When the `notAfter` value of any certificate in the chain is earlier than the current time.
5. Verifies that the server certificate at the head of the chain is suitable for the server with which the client thinks it is communicating.

   > ⓘ **Note:**
   >
   > The client must verify that the address used to connect to the server matches one of the following values:
   >
   > - The `CN` attribute of the certificate's subject.
   > - One of the values of any subject alternative name extension.

These steps represent a starting point. If necessary, the client can perform additional types of validation. For example, if a root or intermediate certification authority maintains a certificate revocation list (CRL) or supports the online certificate status protocol (OCSP), the client must verify that none of the certificates in the chain has been revoked. The client can also verify that the CA certificates include the basic constraints extension, and that the server certificate does not contain too many levels. Other checks, like those that use certificate policy extensions, can also be performed.

Keystores and truststores
A keystore is a type of database that holds certificates.

The following examples represent the most common forms of keystores:

- File that uses the Java-specific Java KeyStore (JKS) format
- File that uses the standard PKCS #12 format
- Collection of files that holds certificates and private keys, typically in PEM or DER format
- Hardware security module (HSM) that makes the certificate information available through an interface like PKCS #11

PingDataSync Server supports file-based keystores by using the JKS and PKCS #12 formats and by using hardware security modules that are accessible through PKCS #11. The server does not currently support a keystore format that consists of individual certificate and private key files. To import these files into a JKS or PKCS #12 keystore, use the **`manage-certificates`** tool.

A keystore also represents a collection of entries, each of which is identified by a name that an alias calls. Keystores can have the following entry types:

Private key entries

Contain a certificate chain and a private key. When a server accepts a TLS-based connection, it uses a private key entry to obtain the certificate chain that it presents to the client. The server can also use the private key from the same entry to process its key agreement. Similarly, a client uses a private key entry when presenting its own certificate chain to a server.

Trusted certificate entries

Contain a single certificate without a private key. As the name implies, a trusted certificate entry is intended primarily for use when determining whether to trust a certificate chain that is presented during TLS negotiation.

Secret key entries

Contain a secret key only, without an associated certificate. These types of entries are not used for TLS processing. Instead, they hold symmetric encryption keys or other types of secrets.

A password, sometimes called a PIN, protects the contents of a keystore. In some cases, like with JKS keystores, some content might be accessible without a password, and a password might be required only when trying to access private keys or secret keys. In other cases, like with PKCS #12 keystores, you might need a password for any interaction with the keystore.

Additional passwords can further protect private keys. This approach is often the same as with the keystore password, but the password can be different. This tactic is useful when a single keystore is shared for multiple purposes, for example, and when merely having access to the keystore does not guarantee access to all of the data that it contains.

> ⓘ **Note:**
>
> A truststore is another name for a keystore that is intended primarily for use when determining whether to trust a certificate chain that has been presented. Truststores generally contain primarily trusted certificate entries, but that case is not required.

Java runtime environments typically include a default truststore, often `jre/lib/security/cacerts` or `lib/security/cacerts`, that is prepopulated with several widely trusted certification authority certificates. When presented with a certificate that one of these authorities has signed, the default truststore can allow the certificate to be trusted without any additional configuration. When presented with a self-signed certificate, or when presented with a certificate that is signed by an issuer not in the default truststore, such as a private corporate certification authority, a separate truststore is required.

Transport Layer Security (TLS)
TLS describes a mechanism for securely communicating between two parties that might have no prior knowledge of each other.

TLS is the successor to SSL, and the two terms are often used interchangeably, even though such usage might not technically be correct.

> ⓘ **Note:**
>
> SSL remains the more widely recognized term. The abbreviation TLS occasionally generates confusion with the StartTLS extended operation, particularly in LDAP.

TLS provides security in the form of the following main components:

Certificate trust

Is about reassuring a connection-initiating client that it is communicating with the server to which it intended to connect. To ensure that the server shares the same degree of confidence in the identity and legitimacy of the client, it can ask the client to present its own certificate chain. For more information, see *Certificate Trust*.

Cipher selection

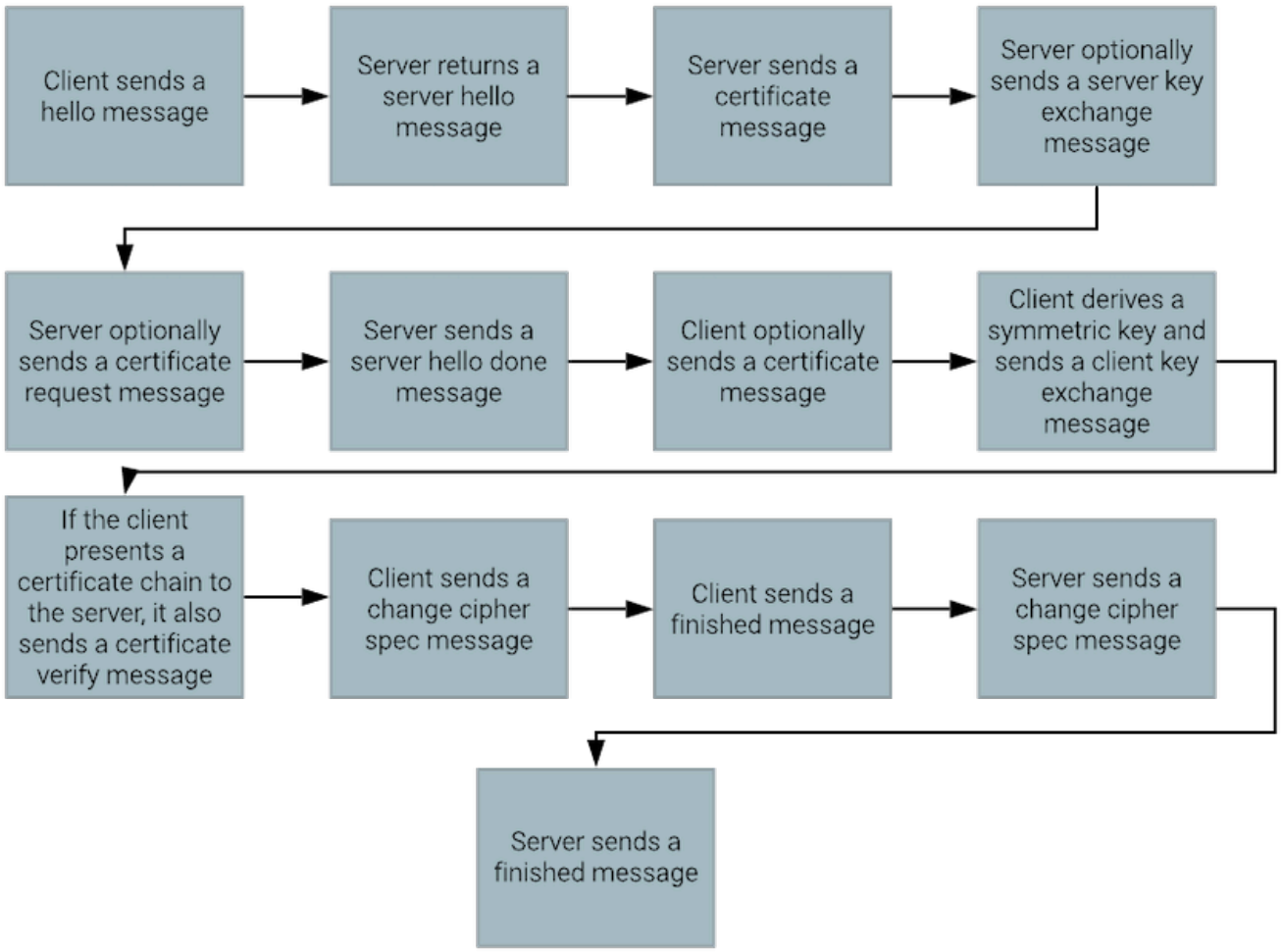
Involves choosing the cipher and the key to protect the bulk of the communication. Although a client can use a server certificate's public key to encrypt data before sending it, this approach can lead to the following issues:

- Unless the client presents its own certificate chain to the server, the server cannot encrypt the data that it sends back to the client.
- Public key encryption is considerably slower than symmetric encryption, in which the same key is used for both encryption and decryption. Public key encryption is also called asymmetric encryption because different keys are used to encrypt and decrypt data.
- If you rely entirely on the security of a private key to ensure the secrecy of a communication, and if the private key becomes compromised, data that has been encrypted with the private key must also be considered compromised.

Rather than relying solely on public key encryption to protect communication between a client and server, the TLS negotiation process allows a client and server to agree on the type of encryption and the secret key to use after completing the negotiation process.

TLS handshakes
The process of negotiating the TLS is referred to as the handshake.

Although the exact process depends on the TLS version that is ultimately chosen, the following steps represent the basic components of a TLS 1.2 handshake:



TLS processing steps

1. The client sends a `hello` message that provides the server with the following information:
   - Highest supported version of the TLS protocol
   - The cipher suites that the client uses
   - Set of extensions with additional information:
     - The address that the client uses to communicate with the server
     - The signature algorithms and elliptic curves that the client supports
     - Whether the client supports secure renegotiation
2. The server returns a `server hello` message that provides the client with the following information:
   - The TLS protocol version that the server uses
   - The cipher suite that the server selects

   The server can also provide its own extensions to the client.
3. The server sends a `certificate` message that provides its certificate chain to the client.
4. The server can optionally send a `server key exchange` message with additional information that the client might need to securely derive the same symmetric encryption key that the server generates.
5. The server can optionally send a `certificate request` message that asks the client to present its own certificate chain to the server.

6. The server sends a `server hello done` message to inform the client that it has completed its `hello` sequence.
7. The client can optionally send a `certificate` message to the server with its own certificate chain.

> ⓘ **Note:**
>
> The client sends a `certificate` message only when the server initially sends a certificate request. If the client receives such a request, it can refuse to, and probably will not, send a certificate chain. The server decides whether to require a client certificate chain. In LDAP, the server commonly asks the client to present a certificate, but continues with TLS negotiation even if the client does not present one. This approach supports authentication methods like SASL EXTERNAL, in which a client uses the certificate chain that it presents during TLS negotiation as proof of its identity.

8. The client derives a symmetric key to use for the remainder of the encrypted processing, and sends a `client key exchange` message to the server. The `client key exchange` message includes the information that the server needs to generate the same key. Only the client and server know the value of the key, even if another entity can observe the communication that passes between the client and the server.
9. If the client presents a certificate chain to the server, it also sends a `certificate verify` message to prove that the private key for the certificate is included at the head of the chain.
10. The client sends a `change cipher spec` message to the server, which informs the server that the client will use the agreed-upon symmetric key to encrypt everything else that it sends to the server.
11. The client sends a `finished` message to the server to indicate that it has completed its portion of the handshake.
12. The server sends a `change cipher spec` message to the client, which informs the client that the server will use the agreed-upon symmetric key to encrypt everything else that it sends to the client.
13. The server sends a `finished` message to the client to indicate that it has completed its portion of the handshake.

TLS 1.3 uses a different handshake sequence that can require only a single round-trip to exchange the necessary information between the client and the server. TLS 1.2 requires two round-trips. To accomplish this task, TLS 1.3 tries to guess the type of key agreement that the server wants to use, and sends the relevant information to the server up front instead of waiting to hear from the server.

Because an extra round of communication between the client and server is eliminated, the server finishes its portion of the negotiation before the client. The server must assume that the client trusts its certificate chain. Because the server might log a successful negotiation only to discover late, through a TLS alert, that the client rejected the certificate, this approach might complicate certain types of troubleshooting.

Key agreement

Key agreement processing provides a critical component of TLS negotiation.

It allows the client and server to select the symmetric key that encrypts the remainder of the communication, but does not reveal the key to anyone who can access the communication. Although several key agreement algorithms are available, the following types are the most common:

RSA key exchange

The client generates random data, uses the server's public key to encrypt it, and provides it to the server, which uses its private key to decrypt it. The client and server alike derive the encryption key from the randomly generated data.

Diffie-Hellman (DH) key exchange

The client and server agree publicly on a pair of mathematically linked numbers, and each participant chooses its own secret value. Through a special computation, they generate a key that can be discovered only by someone who knows one of the secret values. Although several variants of the Diffie-Hellman algorithm can be used in key exchange, we recommend the ECHDE and DHE

versions because they use ephemeral keys with no relation to the server's certificate. Of those two versions, ECDHE is faster and uses smaller keys.

When possible, use ECHDE over DHE, and either of those options over RSA. The DH algorithms provide a substantial benefit over RSA in the form of forward secrecy. Because RSA key exchange uses the server certificate's public key to encrypt data, the encryption can be broken if the certificate's private key is compromised. This warning applies to previously captured data as well as to communication on new TLS connections. The use of ephemeral keys in ECDHE and DHE ensures that, even if the certificate's private key is compromised, the encrypted communication remains indecipherable to anyone but the client and server, although anyone with the private key can still impersonate the legitimate server.

LDAP StartTLS extended operation
In most scenarios, a client that uses TLS establishes a connection to a port that is dedicated to its use, like 636 (LDAPS) or 443 (HTTPS).

The client begins the TLS-negotiation process by sending a `client hello` message over the connection. In some scenarios, the client establishes a non-secure connection and later converts it to a secure one. In LDAP, this task is accomplished by using the **`StartTLS`** extended operation.

The **`StartTLS`** extended operation provides the following advantages over a dedicated LDAPS connection:

- To enable secure as well as insecure communication, only one port needs to be opened through a firewall.
- A client can use opportunistic encryption, in which the client performs the following steps:

  1. Queries the root DSE to determine whether the server supports StartTLS.
  2. Secures the connection, if possible.

  Opportunistic encryption is useful in scenarios like following referrals because LDAP URLs do not officially support LDAPS as a scheme.

To ensure that a communication is always secure, use LDAPS instead of establishing an insecure connection that you secure later with the **`StartTLS`** extended operation. If you enable support for unencrypted LDAP communication, as **`StartTLS`** requires, a client might send a password-containing bind request or other sensitive data over an unencrypted connection. A server can be configured to reject unencrypted communication, but it cannot prevent a client from sending an unencrypted request.

> ⓘ **Note:**
>
> Although you can use **`StartTLS`** to temporarily secure a connection before falling back on an unencrypted LDAP communication, PingDataSync Server does not support this strategy.

About the manage-certificates tool
PingDirectory Server offers a **`manage-certificates`** tool that enables interaction with Java KeyStore (JKS) and PKCS #12 key stores.

Although it behaves similarly to the **`keytool`** utility that accompanies most Java distributions, **`manage-certificates`** is easier to use, provides improved usage information, and offers additional functionality.

Available subcommands

The **`manage-certificates`** tool uses the following subcommands to indicate which function to invoke:

Subcommand	Function
`list-certificates`	Lists the certificates in a keystore.

Subcommand	Function
import-certificate	Imports a certificate into a trusted certificate entry or imports a certificate chain and private key into a private key entry.
export-certificate	Exports a certificate from a keystore.
export-private-key	Exports a private key from a keystore.
generate-self-signed-certificate	Generates a self-signed certificate.
generate-certificate-signing-request	Generates a certificate-signing request that can be provided to a certification authority.
sign-certificate-signing-request	Signs a certificate-signing request with a specified issuer certificate.
check-certificate-usability	Checks a specified certificate in a keystore to verify whether it is suitable for use as a listener certificate.
trust-server-certificate	Initiates the TLS-negotiation process with a specified server to obtain its certificate chain so that a truststore can be updated with the necessary information to trust the chain.
display-certificate-file	Displays the contents of a file that contains one or more PEM-encoded or DER-encoded X.509 certificates.
display-certificate-signing-request-file	Displays the contents of a file that contains a PEM-encoded or DER-encoded PKCS #10 certificate-signing request (CSR).
change-certificate-alias	Changes the alias for an entry in a keystore.
change-keystore-password	Changes the password for a keystore.
change-private-key-password	Changes the password that protects the private key for a specified entry in a keystore.

Common arguments

Most of the **manage-certificates** subcommands require access to a Java KeyStore (JKS) or PKCS #12 keystore. In such instances, use the --keystore argument to specify the path to the keystore.

If the keystore already exists, the tool detects automatically whether it is a JKS or PKCS #12 keystore. If the operation creates a new keystore, you can specify the type explicitly by using the --keystore-type argument, followed by a value of JKS or PKCS12. If you do not specify the keystore type, a default value of JKS is used.

Some situations require you to provide the password that is needed to access the keystore. For a JKS keystore, you might need to provide a keystore password only for operations that involve creating a keystore or accessing a private key. However, you will likely need to provide the password for all operations that involve a PKCS #12 keystore.

To provide a keystore password, use one of the following arguments:

- --keystore-password, followed by the clear-text password for the keystore.

- --keystore-password-file, followed by the path to a file that contains the password for the keystore. The file might contain the password in the clear, or it might be encrypted with a definition from the server's encryption-settings database.
- --prompt-for-keystore-password. If this argument is provided, the tool prompts you interactively to provide the password.

If a private key is protected with a different password than the keystore itself, specify one of the following arguments to provide the private key password:

- --private-key-password, followed by the plaintext password.
- --private-key-password-file, followed by the path to a file that contains the clear-text or encrypted password.
- --prompt-for-private-key-password, which causes the tool to prompt interactively for the password.

Several operations require you to specify the keystore entry to target. In such scenarios, provide the --alias argument, followed by the name of the alias for that entry.

Listing the certificates in a keystore
List the certificates available in a keystore.

Steps

- To list the certificates in a keystore, use the **`list-certificates`** subcommand.

  This subcommand requires you to specify the path to the keystore file, and possibly the password that is needed to access the keystore. The following options are also available:

Option	Description
--alias {alias}	Specifies the alias of the certificate to display. If this value is not provided, all certificates are displayed. To list more than one specific certificate, specify this value multiple times.
--display-pem-certificate	Includes a PEM-encoded representation of each certificate as part of the output.
--verbose	Includes details about each certificate.

  The following command demonstrates the basic listing of a keystore that contains a single certificate chain.

```
$ bin/manage-certificates list-certificates \
     --keystore config/keystore \
     --keystore-password-file config/keystore.pin

Alias:  server-cert (Certificate 1 of 2 in a chain)
Subject DN:  CN=ds1.example.com,O=Example Corp,C=US
Issuer DN:  CN=Example Certification Authority,O=Example Corp,C=US
Validity Start Time:  Saturday, November 9, 2019 at 11:26:09 AM CST
                     (8 minutes, 15 seconds ago)
Validity End Time:  Sunday, November 8, 2020 at 11:26:09 AM CST
(364 days, 23 hours, 51 minutes, 44 seconds from now)
Validity State:  The certificate is currently within the validity window.
Signature Algorithm:  SHA-256 with ECDSA
Public Key Algorithm:  EC (secP256r1)
SHA-1 Fingerprint: 42:f8:85:97:bf:88:bc:74:4b:5b:ce:0c:54:43:9b:44:6b:
                   81:23:a3
SHA-256 Fingerprint: 4f:be:47:ed:36:68:13:38:ba:e8:c0:c5:6c:85:51:97:
                     8b:40:1b:76:10:c0:be:80:15:62:06:96:c5:71:30:df
Private Key Available:  Yes
The certificate has a valid signature.
```

```
Alias:  server-cert (Certificate 2 of 2 in a chain)
Subject DN:  CN=Example Certification Authority,O=Example Corp,C=US
Issuer DN:  CN=Example Certification Authority,O=Example Corp,C=US
Validity Start Time: Saturday, November 9, 2019 at 11:26:08 AM CST
                     (8 minutes, 16 seconds ago)
Validity End Time: Friday, November 4, 2039 at 12:26:08 PM CDT
                   (7299 days, 23 hours, 51 minutes, 43 seconds from now)
Validity State:  The certificate is currently within the validity window.
Signature Algorithm:  SHA-256 with ECDSA
Public Key Algorithm:  EC (secP256r1)
SHA-1 Fingerprint: b8:d0:16:9b:5d:f2:e7:a1:80:79:95:a2:64:b5:aa:ad:80:
                   23:64:16
SHA-256 Fingerprint: cf:98:2a:66:35:6e:6d:f9:5d:25:c6:68:68:04:5a:a8:
                     88:43:ca:b5:c8:e5:c9:95:09:e9:fc:ab:b9:41:ec:71
The certificate has a valid signature.
```

The following sample represents the verbose version of the previous command.

```
$ bin/manage-certificates list-certificates \
     --keystore config/keystore \
     --keystore-password-file config/keystore.pin \
     --verbose

Alias:  server-cert (Certificate 1 of 2 in a chain)
X.509 Certificate Version:  v3
Subject DN:  CN=ds1.example.com,O=Example Corp,C=US
Issuer DN:  CN=Example Certification Authority,O=Example Corp,C=US
Serial Number:  7b:2d:91:6a:ff:51:4f:7a:19:16:26:4f:ce:cb:cb:31
Validity Start Time:  Saturday, November 9, 2019 at 11:26:09 AM CST
(9 minutes, 48 seconds ago)
Validity End Time: Sunday, November 8, 2020 at 11:26:09 AM CST
                   (364 days, 23 hours, 50 minutes, 11 seconds from now)
Validity State:  The certificate is currently within the validity window.
Signature Algorithm:  SHA-256 with ECDSA
Signature Value:
     30:46:02:21:00:cb:d5:5e:45:b2:8a:33:5e:2d:85:23:39:49:d1:3f:8f:dc:
     f8:9e:2f:f3:44:2f:41:0d:69:95:ec:f0:f5:c0:80:02:21:00:ef:8f:32:35:
     3c:88:f4:89:ed:f3:a6:76:
     bb:92:6c:eb:c6:17:ac:61:dc:67:26:f0:ec:67:90:51:28:a1:d0:d5
Public Key Algorithm:  EC (secP256r1)
Elliptic Curve Public Key Is Compressed:  false
Elliptic Curve X-Coordinate:
   -24253153720011259408467676608081666342358203254369897642016197975874105796326
Elliptic Curve Y-Coordinate:
   48722714538591494552787288916186748185323678082126843165293664643134352536146
Certificate Extensions:
     Subject Key Identifier Extension:
          OID:  2.5.29.14
          Is Critical:  false
          Key Identifier:
               21:ad:b9:7a:15:e4:08:13:05:e1:c2:64:0c:86:aa:9b:f0:4c:fb:a0
     Authority Key Identifier Extension:
          OID:  2.5.29.35
          Is Critical:  false
          Key Identifier:
               01:4b:69:99:93:5f:76:51:39:95:61:cc:a9:a8:cb:16:f2:0f:8c:c8
     Subject Alternative Name Extension:
          OID:  2.5.29.17
          Is Critical:  false
          DNS Name:  ds1.example.com
```

```
          DNS Name:  ds.example.com
          DNS Name:  ldap.example.com
          DNS Name:  localhost
          IP Address:  127.0.0.1
          IP Address:  0:0:0:0:0:0:0:1
     Key Usage Extension:
          OID:  2.5.29.15
          Is Critical:  false
          Key Usages:
               Digital Signature
               Key Encipherment
               Key Agreement
     Extended Key Usage Extension:
          OID:  2.5.29.37
          Is Critical:  false
          Key Purpose ID:  TLS Server Authentication
          Key Purpose ID:  TLS Client Authentication
SHA-1 Fingerprint:
   42:f8:85:97:bf:88:bc:74:4b:5b:ce:0c:54:43:9b:44:6b:81:23:a3
SHA-256 Fingerprint:
   4f:be:47:ed:36:68:13:38:ba:e8:c0:c5:6c:85:51:97:8b:40:1b:76:
   10:c0:be:80:15:62:06:96:c5:71:30:df
Private Key Available:  Yes
The certificate has a valid signature.

Alias:  server-cert (Certificate 2 of 2 in a chain)
X.509 Certificate Version:  v3
Subject DN:  CN=Example Certification Authority,O=Example Corp,C=US
Issuer DN:  CN=Example Certification Authority,O=Example Corp,C=US
Serial Number:  43:b7:bb:0c:82:58:42:d8:06:fc:2a:f6:04:e8:2e:8c
Validity Start Time: Saturday, November 9, 2019 at 11:26:08 AM CST
                     (9 minutes, 49 seconds ago)
Validity End Time: Friday, November 4, 2039 at 12:26:08 PM CDT
                   (7299 days, 23 hours, 50 minutes, 10 seconds from now)
Validity State:  The certificate is currently within the validity window.
Signature Algorithm:  SHA-256 with ECDSA
Signature Value:
     30:45:02:21:00:b9:87:50:5d:b7:6a:19:82:99:9b:aa:f1:5d:25:a1:90:3c:
     17:9d:7f:f5:7f:8d:06:b4:57:41:9e:15:c6:5a:af:02:20:0c:00:5e:17:bf:
     ca:bf:0b:ff:db:9f:dc:55:ad:35:eb:df:f6:37:4e:23:83:36:88:d2:cc:
     7d:9e:23:da:78:28
Public Key Algorithm:  EC (secP256r1)
Elliptic Curve Public Key Is Compressed:  false
Elliptic Curve X-Coordinate:

 -20753103001920939059800335367415761738764700353772539765405069978726324039 64
Elliptic Curve Y-Coordinate:

 67079356503908427292378918440889412002659485731683570737365127953554508553 73
Certificate Extensions:
     Subject Key Identifier Extension:
          OID:  2.5.29.14
          Is Critical:  false
          Key Identifier:
               01:4b:69:99:93:5f:76:51:39:95:61:cc:a9:a8:cb:16:f2:0f:8c:c8
     Basic Constraints Extension:
          OID:  2.5.29.19
          Is Critical:  false
          Is CA:  true
          Path Length Constraint:  0
     Key Usage Extension:
          OID:  2.5.29.15
          Is Critical:  false
          Key Usages:
```

```
               Key Cert Sign
               CRL Sign
SHA-1 Fingerprint:
 b8:d0:16:9b:5d:f2:e7:a1:80:79:95:a2:64:b5:aa:ad:80:23:64:16
SHA-256 Fingerprint:

 cf:98:2a:66:35:6e:6d:f9:5d:25:c6:68:68:04:5a:a8:88:43:ca:b5:c8:e5:c9:95:09:
   e9:fc:ab:b9:41:ec:71
The certificate has a valid signature.
```

Generating self-signed certificates

The process of creating a self-signed certificate is straightforward because a self-signed certificate claims itself as its own issuer.

Although self-signed certificates are convenient for testing environments, clients do not trust them by default. Consequently, you should not use them as listener certificates in production environments.

The **`manage-certificates`** tool offers a **`generate-self-signed-certificate`** subcommand that can create a self-signed certificate. In addition to the arguments that provide information about the keystore, certificate alias, and optional private key password, the following arguments are available.

Argument	Description
`--subject-dn {subject}`	Subject DN for the certificate to create. This value is required.
`--days-valid {number}`	Number of days that the certificate remains valid. Defaults to 365 if no value is specified.
`--validity-start-time {timestamp}`	Indicates the time at which the certificate begins its validity window. This value is assumed to reflect the local time zone, and must be expressed in the form YYYYMMDDhhmmss, where a value of 20190102030405 indicates January 2, 2019, at 3:04:05 AM. Defaults to the current time if no value is specified.
`--key-algorithm {name}`	Name of the algorithm to use when generating the key pair. For a listener certificate, this value is typically RSA or EC. Defaults to RSA if no value is specified. **Note:** This argument cannot be used in conjunction with the `--replace-existing-certificate` argument.

Argument	Description
`--key-size-bits {number}`	Length of the key, in bits, to generate. If the `--key-algorithm` argument is given, then --key-size-bits {number} must also be specified. Conversely, if the `--replace-existing-certificate` argument is given, then `--key-size-bits {number}` must not be specified. Typical key sizes are: ▪ RSA key – 2048 or 4096 bits If a default RSA key is used but this argument is not provided, a default key size of 2048 bits is used. ▪ Elliptic curve key – 256 or 384 bits
`--signature-algorithm {name}`	Name of the algorithm to use to sign the certificate. If the `--key-algorithm` argument is used to specify an algorithm other than RSA, then `--signature-algorithm {name}` must also be specified. If the `--replace-existing-certificate` argument is used, then `--signature-algorithm {name}` must not be specified. Typical signature algorithms include SHA256withRSA for certificates with RSA keys, and SHA256withECDSA for certificates with elliptic curve keys. If a default key algorithm is used but the `--signature-algorithm {name}` argument is not provided, a default value of SHA256withRSA is used.
`--replace-existing-certificate`	Uses the new certificate to replace an existing certificate in the key store (within the same alias), and reuses the key for that certificate.
`--inherit-extensions`	Indicates that, when replacing an existing certificate, the new certificate contains the same set of extensions as the existing certificate. If the `--replace-existing-certificate` argument is provided, but the `--inherit-extensions` argument is omitted, the new certificate contains only arguments that are provided explicitly.
`--subject-alternative-name-dns {name}`	Indicates that the certificate is expected to have a subject alternative name extension with the provided DNS name. The given name must be fully qualified, although it can contain an asterisk (*) as a wildcard in the leftmost component. To include multiple DNS names in the subject alternative name extension, specify the `--subject-alternative-name-dns {name}` argument multiple times.

Argument	Description
`--subject-alternative-name-ip-address {address}`	Indicates that the certificate is expected to have a subject alternative name extension with the provided IP address. The given address must be a valid IPv4 or IPv6 address. No wildcards are allowed. To include multiple IP addresses in the subject alternative name extension, specify the `--subject-alternative-name-ip-address {address}` argument multiple times.
`--subject-alternative-name-email-address {address}`	Indicates that the certificate is expected to have a subject alternative name extension with the provided email address. To include multiple email addresses in the subject alternative name extension, specify the `--subject-alternative-name-email-address {address}` argument multiple times.
`--subject-alternative-name-uri {uri}`	Indicates that the certificate is expected to have a subject alternative name extension with the provided URI. To include multiple URIs in the subject alternative name extension, specify the `--subject-alternative-name-uri {uri}` argument multiple times.
`--subject-alternative-name-oid {oid}`	Indicates that the certificate is expected to have a subject alternative name extension with the provided object identifier (OID). The given value must be a valid OID. To include multiple OIDs in the subject alternative name extension, specify the `--subject-alternative-name-oid {oid}` argument multiple times.
`--basic-constraints-is-ca {value}`	Indicates that the certificate is expected to have a basic constraints extension, with a specified value of true or false, for the flag indicating whether to consider the certificate a certification authority that can be used to sign other certificates. ▪ For root and intermediate certificate authority (CA) certificates, the `--basic-constraints-is-ca {value}` argument must be present with a value of true. ▪ For end-entity certificates, the `--basic-constraints-is-ca {value}` argument can optionally be present with a value of false. ▪ For a self-signed certificate, specify the `--basic-constraints-is-ca {value}` argument with a value of false to indicate that the certificate is not considered a CA certificate.

Argument	Description
`--basic-constraints-maximum-path-length {number}`	Indicates that the basic constraints extension is expected to include a path length constraint element with the specified value. Use this argument only if `--basic-constraints-is-ca` is provided with a value of true. A path length constraint value of 0 indicates that the certificate can be used to issue only end-entity certificates. A path length constraint value of 1 indicates that the certificate can be used to sign end-entity certificates or intermediate CA certificates, the latter of which can be used to sign only end-entity certificates. A value greater than 1 indicates the presence of several intermediate CA certificates between it and the end-entity certificate at the head of the chain.
`--key-usage {value}`	Indicates that the certificate is expected to have a key usage extension with the specified value. The following values are allowed: ▪ digital-signature ▪ non-repudiation ▪ key-encipherment ▪ data-encipherment ▪ key-agreement ▪ key-cert-sign ▪ crl-sign ▪ encipher-only ▪ decipher-only To include multiple key usages, specify the `--key-usage {value}` argument multiple times.
`--extended-key-usage {value}`	Indicates that the certificate is expected to have an extended key usage extension with the specified value. The following values are allowed: ▪ server-auth ▪ client-auth ▪ code-signing ▪ email-protection ▪ time-stamping ▪ ocsp-signing

For example, the following command can be used to generate a self-signed server certificate.

```
bin/manage-certificates generate-self-signed-certificate \
     --keystore config/keystore \
     --keystore-password-file config/keystore.pin \
     --keystore-type JKS \
     --alias server-cert \
```

```
     --subject-dn "CN=ds.example.com,O=Example Corp,C=US" \
     --key-algorithm EC \
     --key-length-bits 256 \
     --signature-algorithm SHA256withECDSA \
     --subject-alternative-name-dns ds.example.com \
     --subject-alternative-name-dns ds1.example.com \
     --subject-alternative-name-dns localhost \
     --subject-alternative-name-ip-address 1.2.3.4 \
     --subject-alternative-name-ip-address 127.0.0.1 \
     --subject-alternative-name-ip-address 0:0:0:0:0:0:0:1 \
     --key-usage digital-signature \
     --key-usage key-encipherment \
     --key-usage key-agreement \
     --extended-key-usage server-auth \
     --extended-key-usage client-auth

Successfully created a new JKS keystore.

Successfully generated the following self-signed certificate:
Subject DN:  CN=ds.example.com,O=Example Corp,C=US
Issuer DN:  CN=ds.example.com,O=Example Corp,C=US
Validity Start Time: Monday, January 27, 2020 at 03:40:13 PM
 CST
                     (0 seconds ago)
Validity End Time: Tuesday, January 26, 2021 at 03:40:13 PM CST
                   (364 days, 23 hours, 59 minutes, 59 seconds
 from now)
Validity State:  The certificate is currently within the
 validity window.
Signature Algorithm:  SHA-256 with ECDSA
Public Key Algorithm:  EC (secP256r1)
SHA-1 Fingerprint:
 4f:41:82:7f:08:e9:d8:05:8c:19:8b:3e:5b:bc:59:98:d3:15:71:3a
SHA-256 Fingerprint:

 76:e6:8e:c5:c8:8d:27:ce:2b:85:b9:8c:9d:49:3c:06:f4:40:f1:d0:70:67:39:24:fc:
   31:bc:f8:51:83:f2:42
```

Generating certificate signing requests

A certificate signing request (CSR) contains all of the information that a certification authority requires to issue a certificate.

RFC 2986 defines the request format, also known as PKCS #10, and includes the following elements:

- Certificate signing request version
- Requested subject distinguished name (DN) for the certificate
- Public key for the requested certificate
- Requested set of extensions for the certificate
- Signature that proves the requester has the private key for the given public key

To create a certificate signing request, use the **`manage-certificates generate-certificate-signing-request`** command, which performs the following steps:

1. Generated a public and private key pair.
2. Stores the key pair in a key store with a given alias.
3. Outputs the certificate signing request to the terminal.
4. Optionally writes the certificate signing request to a file.

Because a certificate signing request contains many of the same elements as a certificate, the command to generate one takes most of the same arguments as for generating a self-signed certificate. The following arguments are unavailable when generating a CSR:

- `--replace-existing-certificate`
- `--days-valid {number}`
- `--validity-start-time {timestamp}`

The following arguments are available when generating a certificate signing request but not when generating a self-signed certificate:

`--output-file {path}`

Path to a file to which the certificate signing request is written. If this value is not provided, the request is written only to the terminal in PEM form.

`--output-format {value}`

Format to use when writing the certificate signing request. This value can be PEM or DER, but the DER format is used only in conjunction with the `--output-file` argument. Defaults to PEM if the `--output-format {value}` argument is not provided.

`--use-existing-key-pair`

Indicates that the CSR uses a key pair that already exists in the key store with the given alias, rather than generating a new key pair, in which case the specified alias must not already be in use in the key store.

The following example command creates a CSR.

```
bin/manage-certificates generate-certificate-signing-request \
     --output-file ds1-cert.csr \
     --output-format PEM \
     --keystore config/keystore \
     --keystore-password-file config/keystore.pin \
     --keystore-type JKS \
     --alias server-cert \
     --subject-dn "CN=ds.example.com,O=Example Corp,C=US" \
     --key-algorithm EC \
     --key-length-bits 256 \
     --signature-algorithm SHA256withECDSA \
     --subject-alternative-name-dns ds.example.com \
     --subject-alternative-name-dns ds1.example.com \
     --subject-alternative-name-dns localhost \
     --subject-alternative-name-ip-address 1.2.3.4 \
     --subject-alternative-name-ip-address 127.0.0.1 \
     --subject-alternative-name-ip-address 0:0:0:0:0:0:0:1 \
     --key-usage digital-signature \
     --key-usage key-encipherment \
     --key-usage key-agreement \
     --extended-key-usage server-auth \
     --extended-key-usage client-auth

Successfully created a new JKS keystore.

Successfully generated the key pair to use for the certificate signing
request.

Successfully wrote the certificate signing request to file
'/ds/build/package/PingDirectory/ds1-cert.csr'.
```

If the contents of the resulting CSR file are made available to a certification authority to be signed, the resulting signed certificate can be imported into the key store.

To print the contents of a certificate signing request file, use the **`display-certificate-signing-request-file`** subcommand, which supports the following arguments:

--certificate-signing-request-file {path}

Path to the file that contains the certificate signing request to display.

--verbose

Indicates that the command is expected to display verbose information about the request, rather than a basic information set.

The following example demonstrates the basic output from the command.

```
$ bin/manage-certificates display-certificate-signing-request-file \
    --certificate-signing-request-file ds1-cert.csr

    PKCS #10 Certificate Signing Request Version:  v1
    Subject DN:  CN=ds.example.com,O=Example Corp,C=US
    Signature Algorithm:  SHA-256 with ECDSA
    Public Key Algorithm:  EC (secP256r1)
```

The following example demonstrates the verbose output.

```
$ bin/manage-certificates display-certificate-signing-request-file \
     --certificate-signing-request-file ds1-cert.csr \
     --verbose

PKCS #10 Certificate Signing Request Version:  v1
Subject DN:  CN=ds.example.com,O=Example Corp,C=US
Signature Algorithm:  SHA-256 with ECDSA
Signature Value:

 30:45:02:20:46:31:be:9e:6d:2f:0e:e3:d0:80:5c:88:ef:da:86:07:fd:15:b7:62:
     83:45:39:0a:c9:f2:f9:17:eb:08:94:ff:02:21:00:c8:bd:df:57:fa:ea:8c:04:
     df:c5:27:76:e5:b3:3b:4f:df:ec:d3:e4:09:5b:c0:6c:7b:86:39:ec:d0:0e:c1:64
Public Key Algorithm:  EC (secP256r1)
Elliptic Curve Public Key Is Compressed:  false
Elliptic Curve X-Coordinate:
   208628537904757963197889471667098239762296638799662436502070112279302 4
   3221133
Elliptic Curve Y-Coordinate:
   479697739226644990505743464941788269420922508654777168408919906254139
   60212095
Certificate Extensions:
     Subject Key Identifier Extension:
          OID:  2.5.29.14
          Is Critical:  false
          Key Identifier:
               f2:de:fd:bf:d3:2f:96:ef:01:70:2d:0e:85:f5:fb:17:d5:a0:9e:67
     Subject Alternative Name Extension:
          OID:  2.5.29.17
          Is Critical:  false
          DNS Name:  ds.example.com
          DNS Name:  ds1.example.com
          DNS Name:  localhost
          IP Address:  1.2.3.4
          IP Address:  127.0.0.1
          IP Address:  0:0:0:0:0:0:0:1
     Key Usage Extension:
          OID:  2.5.29.15
          Is Critical:  false
          Key Usages:
               Digital Signature
               Key Encipherment
```

```
               Key Agreement
     Extended Key Usage Extension:
          OID:  2.5.29.37
          Is Critical:  false
          Key Purpose ID:  TLS Server Authentication
          Key Purpose ID:  TLS Client Authentication
```

Importing signed and trusted certificates

Use the **`manage-certificates import-certificate`** command to import certificates into a keystore.

This command is used to accomplish the following tasks:

- Import a certificate that a certification authority has signed into the keystore in which the key pair was generated. In this scenario, the certificate is imported into a private key entry and must be imported as a certificate chain rather than an end-entity certificate.
- Import a trusted issuer certificate into a trust store. In this scenario, the certificate is imported into a trusted certificate entry as a single certificate instead of as a chain.
- Import a certificate chain, along with the private key for the end-entity certificate. This approach imports certificates that were generated through another library, like OpenSSL.

In addition to the arguments that provide information about the key store and the alias into which the certificate or certificate chain is imported, the **`manage-certificates import-certificate`** command accepts the following arguments:

`--certificate-file {path}`

Path to the file that contains the certificate to import. The certificate can be in PEM or DER format and can be a single certificate or a certificate chain. If the certificates in the chain reside in separate files, specify the --certificate-file {path} argument multiple times when you import a certificate chain.

`--private-key-file {path}`

Path to the file containing the private key that corresponds to the certificate at the head of the imported chain. The private key can be in PEM or DER format.

`--no-prompt`

Indicates that the certificate is to be imported without prompting for confirmation. By default, a summary of the certificate is displayed, and you must confirm that you want to import it.

The following example command imports a signed certificate into the key store that generates the certificate signing request.

```
$ bin/manage-certificates import-certificate \
     --keystore config/keystore \
     --keystore-password-file config/keystore.pin \
     --alias server-cert \
     --certificate-file ds1-cert.pem \
     --certificate-file ca-cert.pem

The following certificate chain will be imported into the keystore into
 alias
'server-cert', preserving the existing private key associated with that
 alias:

Subject DN:  CN=ds.example.com,O=Example Corp,C=US
Issuer DN:  CN=Example Root CA,O=Example Corp,C=US
Validity Start Time: Sunday, November 10, 2019 at 09:09:23 PM CST
                     (4 minutes, 16 seconds ago)
Validity End Time: Monday, November 9, 2020 at 09:09:23 PM CST
                   (364 days, 23 hours, 55 minutes, 43 seconds from now)
Validity State:  The certificate is currently within the validity window.
```

```
Signature Algorithm:  SHA-256 with ECDSA
Public Key Algorithm:  EC (secP256r1)
SHA-1 Fingerprint:
 02:51:25:43:3e:68:f5:71:36:e3:5d:df:74:de:f6:a1:5a:db:0f:eb
SHA-256 Fingerprint: 1d:b5:eb:3c:f5:ff:bf:79:a2:a5:86:b8:e4:33:76:4d:d7:
                     50:dc:a4:34:95:37:be:89:45:86:1f:5d:79:c3:93

Subject DN:  CN=Example Root CA,O=Example Corp,C=US
Issuer DN:  CN=Example Root CA,O=Example Corp,C=US
Validity Start Time: Sunday, November 10, 2019 at 09:00:07 PM CST
                     (13 minutes, 32 seconds ago)
Validity End Time: Saturday, November 5, 2039 at 10:00:07 PM CDT
                   (7299 days, 23 hours, 46 minutes, 27 seconds from now)
Validity State:  The certificate is currently within the validity window.
Signature Algorithm:  SHA-256 with ECDSA
Public Key Algorithm:  EC (secP384r1)
SHA-1 Fingerprint:
 0e:5c:21:c9:a5:36:0a:24:eb:aa:55:b6:a5:94:0e:e0:56:03:22:e6
SHA-256 Fingerprint:
 77:cf:66:d7:3c:8a:fd:67:2d:b7:36:fd:60:1d:ca:eb:1b:03:b1:
                     12:7b:10:1f:26:05:b7:b9:0d:02:e0:38:3e

Do you want to import this certificate chain into the keystore? yes

Successfully imported the certificate chain.
```

If you do not provide the `--no-prompt` argument, the **`manage-certificates import-certificate`** tool still displays information about the certificates to import. To view additional information about a certificate before you import it, use the **`display-certificate-file`** subcommand, which supports the following arguments:

`--certificate-file {path}`

Path to the file that contains the certificate to view.

`--verbose`

Displays verbose information about the certificate.

The output of the **`display-certificate-file`** subcommand has the same format and content as the **`list-certificates`** subcommand.

Exporting certificates

Use the **`export-certificate`** subcommand to export a single certificate or a certificate chain from a key store to a file in PEM or DER format.

The **`export-certificate`** subcommand supports the normal arguments about the key store and certificate alias, in addition to the following arguments:

`--output-file {path}`

Path to the file to which exported certificates are written. If this value is not provided, the certificates are written to standard output rather than a file.

`--output-format {format}`

Format in which exported certificates are written. The value can be PEM or DER, but the DER format can be used only if the output is written to a file. Defaults to PEM if no value is specified.

`--export-certificate-chain`

Indicates that a certificate chain, rather than the end-entity certificate only, is to be exported.

`--separate-file-per-certificate`

Indicates the use of separate output files for each exported certificate, rather than placing all of the certificates in a single file. If this argument is provided and multiple certificates are to be exported, then .1 is appended to the path for the indicated output file for the first certificate in the chain, .2 is appended for the second certificate, and so on.

The following example exports a certificate chain.

```
$ bin/manage-certificates export-certificate \
     --keystore config/keystore \
     --keystore-password-file config/keystore.pin \
     --alias server-cert \
     --output-file server-cert.pem \
     --output-format PEM \
     --export-certificate-chain \
     --separate-file-per-certificate

Successfully exported the following certificate to '/ds/server-cert.pem.1':
Subject DN:  CN=ds.example.com,O=Example Corp,C=US
Issuer DN:  CN=Example Root CA,O=Example Corp,C=US
Validity Start Time: Sunday, November 10, 2019 at 09:09:23 PM CST
                     (3 hours, 26 minutes, 23 seconds ago)
Validity End Time: Monday, November 9, 2020 at 09:09:23 PM CST
                           (364 days, 20 hours, 33 minutes, 36 seconds from
 now)
Validity State:  The certificate is currently within the validity window.
Signature Algorithm:  SHA-256 with ECDSA
Public Key Algorithm:  EC (secP256r1)
SHA-1 Fingerprint:
 02:51:25:43:3e:68:f5:71:36:e3:5d:df:74:de:f6:a1:5a:db:0f:eb
SHA-256 Fingerprint:
1d:b5:eb:3c:f5:ff:bf:79:a2:a5:86:b8:e4:33:76:4d:d7:50:dc:a4:34:95:37:be:89:45:
86:1f:5d:79:c3:93

Successfully exported the following certificate to '/ds/server-cert.pem.2':
Subject DN:  CN=Example Root CA,O=Example Corp,C=US
Issuer DN:  CN=Example Root CA,O=Example Corp,C=US
Validity Start Time: Sunday, November 10, 2019 at 09:00:07 PM CST
                     (3 hours, 35 minutes, 39 seconds ago)
Validity End Time: Saturday, November 5, 2039 at 10:00:07 PM CDT
                   (7299 days, 20 hours, 24 minutes, 20 seconds from now)
Validity State:  The certificate is currently within the validity window.
Signature Algorithm:  SHA-256 with ECDSA
Public Key Algorithm:  EC (secP384r1)
SHA-1 Fingerprint:
 0e:5c:21:c9:a5:36:0a:24:eb:aa:55:b6:a5:94:0e:e0:56:03:22:e6
SHA-256 Fingerprint:
   77:cf:66:d7:3c:8a:fd:67:2d:b7:36:fd:60:1d:ca:eb:1b:03:b1:12:7b:10:1f:26:
   05:b7:b9:0d:02:e0:38:3e
```

The **`export-certificate`** subcommand exports only the public portion of a certificate. Its private key is not included. To export the private key, use the **`export-private-key`** subcommand, which supports the following arguments, in addition to the usual key store and alias arguments:

`--output-file {path}`

Path to the file to which the exported private key is written. If this value is not provided, the key is written to standard output rather than a file.

`--output-format {format}`

Format in which the exported private key is written. The value can be PEM or DER, but the DER format is used only if the output is written to a file. Defaults to PEM if no value is specified.

The following code provides an example of the **`export-private-key`** subcommand .

```
$ bin/manage-certificates export-private-key \
     --keystore config/keystore \
     --keystore-password-file config/keystore.pin \
     --alias server-cert \
     --output-file server-cert-key.pem \
     --output-format PEM

Successfully exported the private key.
```

Using manage-certificates as a simple certification authority

If your PingDataSync Server instances need to support an arbitrary or unknown set of clients, configure them with certificates from a trusted issuer, such as a commercial authority or the free Let's Encrypt service.

About this task

If you control every client that accesses the servers, you might want to create your own internal certification authority so that you have a common issuer for all servers. In such a scenario, the clients need to trust only the certificates that the issuer signs. Commercial and open-source software packages provide full-featured certification authority functionality, but you can use the **`manage-certificates`** tool to create a certificate authority (CA) certificate that you can use to sign certificate-signing requests.

Steps

1. Create a CA certificate.

   A CA certificate is a self-signed certificate that possesses the following extensions:

   - A key usage extension that includes at least the keyCertSign usage
   - A basic constraints extension that identifies the certificate as a CA certificate

   If you do not plan to use an intermediate CA certificate, the basic constraints extension must have a path length constraint of 0. If you plan to use an intermediate CA certificate, the path length constraint must be 1. Because certificates that the CA certificate signs are valid only for as long as all certificates in the chain remain valid, we recommend that you specify a long lifespan for the CA certificate.

   The following example creates a new root CA certificate.

   ```
   $ bin/manage-certificates generate-self-signed-certificate \
        --keystore /ca/root-ca-keystore \
        --keystore-password-file /ca/root-ca-keystore.pin \
        --keystore-type JKS \
        --alias root-ca-cert \
        --subject-dn "CN=Example Root CA,O=Example Corp,C=US" \
        --days-valid 7300 \
        --key-algorithm RSA \
        --key-size-bits 4096 \
        --signature-algorithm SHA256withRSA \
        --basic-constraints-is-ca true \
        --basic-constraints-maximum-path-length 1 \
        --key-usage key-cert-sign \
        --key-usage crl-sign

   Successfully created a new JKS keystore.

   Successfully generated the following self-signed certificate:
   Subject DN:  CN=Example Root CA,O=Example Corp,C=US
   Issuer DN:  CN=Example Root CA,O=Example Corp,C=US
   Validity Start Time: Monday, January 27, 2020 at 03:47:29 PM CST (0
    seconds ago)
   ```

```
Validity End Time: Sunday, January 22, 2040 at 03:47:29 PM CST
                   (7299 days, 23 hours, 59 minutes, 59 seconds from now)
Validity State:  The certificate is currently within the validity window.
Signature Algorithm: SHA-256 with RSA
Public Key Algorithm: RSA (4096-bit)
SHA-1 Fingerprint:
 bc:8e:5b:30:52:ec:03:63:b4:9a:aa:1a:45:a0:fc:84:49:dd:e8:64
SHA-256 Fingerprint:

 d5:47:06:cd:a2:95:42:61:1f:c7:aa:04:16:1e:c1:70:41:c4:44:48:bf:74:20:5f:1c:
   61:e2:aa:40:08:3a:ff
```

2. Export the public portion of the root CA certificate for future reference.

   When you import a signed certificate, you can import the public portion of the root CA certificate as a standalone certificate into trust stores as well as into part of a certificate chain.

3. Create a new certificate signing request to create an intermediate CA certificate,

   The certificate signing request uses essentially the same settings as the root CA. If you anticipate only a single intermediate CA, its basic constraints extension must have a path length constraint of 0, rather than 1, to indicate that it is used only to sign end-entity certificates and that it cannot create subordinate CA certificates by itself.

   The following example command creates a certificate signing request.

```
$ bin/manage-certificates generate-certificate-signing-request \
     --keystore /ca/intermediate-ca-keystore \
     --keystore-password-file /ca/intermediate-ca-keystore.pin \
     --keystore-type JKS \
     --alias intermediate-ca-cert \
     --subject-dn "CN=Example Intermediate CA,O=Example Corp,C=US" \
     --key-algorithm RSA \
     --key-size-bits 4096 \
     --signature-algorithm SHA256withRSA \
     --basic-constraints-is-ca true \
     --basic-constraints-maximum-path-length 0 \
     --key-usage key-cert-sign \
     --key-usage crl-sign \
     --output-file /ca/intermediate-ca-cert.csr \
     --output-format PEM

Successfully created a new JKS keystore.

Successfully generated the key pair to use for the certificate signing
request.

Successfully wrote the certificate signing request to file
'/ca/intermediate-ca-cert.csr'.
```

4. Use the root CA certificate to sign the certificate signing request for the intermediate CA certificate with the **`sign-certificate-signing-request`** subcommand.

   The **`sign-certificate-signing-request`** subcommand takes most of the same arguments as generating a self-signed certificate. The primary differences between the argument sets are as follows:

   - The key store that contains the certificate uses the provided key store arguments to sign the request. To specify the name of the certificate to use when signing the request, use the `--signing-certificate-alias` argument.
   - To specify the path to the file that contains the certificate signing request file to generate, provide a `--request-input-file` argument.
   - To specify the path to the file to which the signed certificate is written, provide a `--certificate-output-file` argument. If this argument is omitted, the PEM representation of the certificate is written to standard output.
   - To specify the format, PEM or DER, in which the certificate is written to the output file, provide an `--output-format` argument.
   - To specify the subject to use for the signed certificate, use the `--subject-dn` argument. To use the subject DN from the certificate signing request, omit this argument.
   - To specify the name of the signature algorithm, use the `--signature-algorithm` argument.

     > ⓘ **Note:**
     >
     > Because the requester generated the key, you cannot specify the key algorithm or the key length.

   - To indicate that the signed certificate includes every extension that is listed in the certificate signing request, use the `--include-requested-extensions` argument. If this argument is not provided, explicitly specify the set of extensions to include.

   The following example command signs the certificate signing request for an intermediate CA certificate.

   ```
   $ bin/manage-certificates sign-certificate-signing-request \
        --keystore /ca/root-ca-keystore \
        --keystore-password-file /ca/root-ca-keystore.pin \
        --signing-certificate-alias root-ca-cert \
        --days-valid 7300 \
        --include-requested-extensions \
        --request-input-file /ca/intermediate-ca-cert.csr \
        --certificate-output-file /ca/intermediate-ca-cert.pem \
        --output-format PEM

   Read the following certificate signing request:

   PKCS #10 Certificate Signing Request Version:  v1
   Subject DN:  CN=Example Intermediate CA,O=Example Corp,C=US
   Signature Algorithm:  SHA-256 with RSA
   Public Key Algorithm:  RSA (4096-bit)

   Do you really want to sign this request? yes

   Successfully wrote the signed certificate to file
   '/ca/intermediate-ca-cert.pem'.
   ```

5. After you obtain the intermediate CA certificate, create secure, offline backups of the root CA certificate.

6. Remove the root CA certificate, or at least its private key, from all systems.

> ⓘ **Note:**
>
> Make certain that all end-entity certificates are signed with the intermediate CA certificate, and that the process is identical to the previous example. Restore the root CA certificate only if you need to sign another intermediate CA certificate.

Enabling TLS support during setup
Enable TLS support in the server.

To enable TLS support in the server, you should complete one of the following tasks during the setup procedure:

- Provide a key store that contains the certificate to use.
- Make the installer generate a self-signed certificate.

When using the **`setup`** tool in interactive mode, it prompts you for the information that it needs to configure secure communication, as shown in the following example.

```
Do you want to enable the Directory Server services (Available State,
Available or Degraded State, Configuration, Consent, Directory REST API,
Documentation, SCIM2, and Swagger UI) and Administrative Console over
HTTPS?  After setup, you can selectively enable or disable individual
services and applications by configuring the HTTPS Connection Handler
(yes / no) [yes]: yes

On which port should the Directory Server accept connections from HTTPS
clients? [443]: 443

On which port should the Directory Server accept connections from LDAP
clients? [389]: 389

Do you want to enable LDAPS? (yes / no) [yes]: yes
On which port should the Directory Server accept connections from LDAPS
clients? [636]: 636

Do you want to enable StartTLS? (yes / no) [yes]: yes

Certificate server options:

    1)  Generate self-signed certificate (recommended for testing purposes
        only)
    2)  Use an existing certificate located on a Java Keystore (JKS)
    3)  Use an existing certificate located on a PKCS12 keystore
    4)  Use an existing certificate on a PKCS11 token

Enter option [1]: 2

Java Keystore (JKS) path: /ca/ds1-keystore
Keystore PIN: {password}

Truststore options:

    1)  Generate a default JKS truststore
    2)  Use an existing JKS truststore
    3)  Use an existing PKCS12 truststore

Enter option [1]: 2

JKS truststore path: /ca/truststore
```

```
Truststore password (can be blank): {password}
```

When using **setup** in non-interactive mode, use the following arguments to configure TLS support.

Argument	Description
`--ldapsPort {port}`	Server enables support for LDAPS (LDAP over TLS) on the specified TCP port.
`--httpsPort {port}`	Server enables support for HTTPS for SCIM, the Directory REST API, and the web-based administration console on the specified TCP port.
`--enableStartTLS`	LDAP connection handler enables support for the StartTLS extended operation.
`--generateSelfSignedCertificate`	**setup** generates a self-signed certificate that is presented to clients that use LDAPS, HTTPS, and the StartTLS extended operation.
`--useJavaKeyStore {path}`	Server uses the specified Java KeyStore (JKS) key store to obtain the certificate chain that it presents to clients that use LDAPS, HTTPS, and the StartTLS extended operation.
`--usePKCS12KeyStore {path}`	Server uses the specified PKCS #12 key store to obtain the certificate chain that it presents to clients that use LDAPS, HTTPS, and the StartTLS extended operation.
`--usePKCS11KeyStore`	Server uses a PKCS #11 key store, like a hardware security module, to obtain the certificate chain that it presents to clients that use LDAPS, HTTPS, and the StartTLS extended operation. The Java Virtual Machine (JVM) must already be configured to access the appropriate key store through PKCS #11.
`--keyStorePassword {password}`	Password that is needed to interact with the specified JKS, PKCS #12, or PKCS #11 key store. The **setup** tool assumes that the private key password matches the key store password.
`--keyStorePasswordFile {path}`	Path to the file that contains the password needed to interact with the specified JKS, PKCS #12, or PKCS #11 key store.
`--certNickname {alias}`	Alias of the private key entry in the specified key store that contains the certificate chain to present to clients during TLS negotiation. This argument is optional but recommended if the key store contains multiple certificates.
`--useJavaTrustStore {path}`	Server uses the specified JKS trust store to determine whether to trust certificate chains that are presented to it during TLS negotiation.

Argument	Description
`--usePKCS12TrustStore {path}`	Server uses the specified PKCS #12 trust store to determine whether to trust certificate chains that are presented to it during TLS negotiation
`--trustStorePassword {password}`	Password that is needed to interact with the specified JKS or PKCS #11 trust store.
`--trustStorePasswordFile {path}`	Path to the file that contains the password needed to interact with the specified JKS or PKCS #11 trust store.

The following example command sets up PingDirectory Server in non-interactive mode with an existing certificate.

```
$ ./setup \
     --no-prompt \
     --acceptLicense \
     --ldapPort 389 \
     --ldapsPort 636 \
     --httpsPort 443 \
     --enableStartTLS \
     --useJavaKeyStore config/keystore \
     --keyStorePasswordFile config/keystore.pin \
     --certNickname server-cert \
     --useJavaTrustStore config/truststore \
     --trustStorePasswordFile config/truststore.pin \
     --baseDN dc=example,dc=com \
     --rootUserDN "cn=Directory Manager" \
     --rootUserPasswordFile root-pw.txt \
     --maxHeapSize 10g \
     --encryptDataWithPassphraseFromFile encryption-settings-password.txt \
     --instanceName ds1 \
     --location Austin \
     --noPropertiesFile


Ping Identity Directory Server 8.0.0.0

Initializing ..... Done
Configuring Directory Server ..... Done
Configuring Certificates ..... Done
Starting Directory Server ..... Done

Access product documentation from docs/index.html
```

Enabling TLS support after setup

If the server has been set up without support for TLS, enable TLS support later by completing the following tasks.

Steps

1. Obtain a certificate chain.

   For more information about obtaining a certificate chain, see *Certificate chains* on page 322. To prepare a Java KeyStore JKS or PKCS #12 key store with an appropriate certificate chain and private key, use the **`manage-certificates`** tool. We also recommend that you create a trust store that the server can use.

2. Configure the key and trust manager providers.

   For more information, see *Configuring key and trust manager providers* on page 350.
3. Configure connection handlers.

   For more information, see *Configuring connection handlers* on page 350.

Configuring key and trust manager providers
After you have a key store, configure a key manager provider to access it.

The server is preconfigured with key manager providers, `JKS` and `PKCS12`, that you can use with JKS or PKCS #12 key stores, respectively. You can update the appropriate key manager provider in most cases to reference the key store that you plan to use. The following code provides an example.

```
dsconfig set-key-manager-provider-prop \
     --provider-name JKS \
     --set enabled:true \
     --set key-store-file:config/keystore \
     --set key-store-pin-file:config/keystore.pin
```

A similar change configures a trust manager provider to reference the appropriate trust store. The following code provides an example.

```
dsconfig set-trust-manager-provider-prop \
     --provider-name JKS \
     --set enabled:true \
     --set include-jvm-default-issuers:true \
     --set trust-store-file:config/truststore
     --set trust-store-pin-file:config/truststore.pin
```

ⓘ **Note:**

If all clients and servers use certificates that are signed by issuers and are included in the JVM's default trust store, you can use the `JVM-Default` trust manager provider to accomplish this task.

Configuring connection handlers
After you configure the key and trust manager providers, update the connection handlers to use the key and trust manager providers.

Steps

- For the LDAP connection handler, use the following command to enable StartTLS with a configuration change. By default, the LDAP connection handler accepts non-secure connections.

  ```
  dsconfig set-connection-handler-prop \
    --handler-name "LDAP Connection Handler" \
    --set allow-start-tls:true \
    --set key-manager-provider:JKS \
    --set trust-manager-provider:JKS \
    --set ssl-cert-nickname:server-cert \
    --set ssl-client-auth-policy:optional
  ```
- If you did not configure secure communication during setup, the LDAPS connection handler is disabled. To configure LDAPS support in this scenario, enable the connection handler and configure most of the same settings. You must set allow-start-tls to `false` and use-ssl to `true`. See the following code for an example configuration.

  ```
  dsconfig set-connection-handler-prop \
    --handler-name "LDAPS Connection Handler" \
    --set enabled:true \
  ```

```
  --set key-manager-provider:JKS \
  --set trust-manager-provider:JKS \
  --set ssl-cert-nickname:server-cert \
  --set ssl-client-auth-policy:optional
```

The following example uses a similar configuration change to enable the HTTPS connection handler.

```
dsconfig set-connection-handler-prop \
  --handler-name "HTTPS Connection Handler" \
  --set enabled:true \
  --set listen-port:443 \
  --set key-manager-provider:JKS \
  --set trust-manager-provider:JKS \
  --set ssl-cert-nickname:server-cert
```

Updating the topology registry

After the server connection handlers are updated to enable TLS, update the topology registry to provide information about the new configuration.

The topology registry holds information about server instances that are part of the environment, and it helps to facilitate inter-server communication, such as replication, mirroring portions of the configuration, and PingDirectory Server's automatic backend server-discovery functionality.

The following table details the two types of entries that require updating.

Configuration types and their update descriptions

Configuration Type	Update description

Server instance listener configuration	▪ Provides information that is needed to trust the TLS certificates that instances in the topology present.
	▪ The server instance listener configuration must include the server certificate, which is defined as the certificate at the head of the chain. This version must be the multi-line, PEM-formatted representation of the certificate. You can use **`dsconfig`** to import the certificate from a file, as shown in the following example.
	`bin/dsconfig set-server-instance-listener-prop \`
	`  --instance-name ds1 \`
	`  --listener-name ldap-listener-mirrored-config \`
	`  --set server-ldap-port:636 \`
	`  --set connection-security:ssl \`
	`  --set 'listener-certificate>/ca/ds1-cert.pem'`
	Note:
	The less-than operator > in the final line indicates that the value is read from a file rather than provided directly. In addition, you might not need to enclose the property name and path within single straight quotes to prevent the shell from interpreting the less-than symbol as an attempt to redirect input.

Server instance configuration	▪ Provides information about options for communicating with those instances.
	▪ Update the server instance configuration object to reflect the new methods that are available for communication with the instance. For example, the `preferred-security` property identifies the mechanism by which other instances in the topology attempt to communicate with the instance.
	The following example code sets the LDAPS and HTTPS ports, indicates that StartTLS support is enabled, and instructs other instances to use SSL (LDAPS) when communicating with the instance.
	`dsconfig set-server-instance-prop \`
	`  --instance-name ds1 \`
	`  --set ldaps-port:636 \`
	`  --set https-port:443 \`
	`  --set preferred-security:ssl \`
	`  --set start-tls-enabled:true`

Troubleshooting TLS-related issues

Use this section for troubleshooting problems that might arise during TLS configuration, including communication and security issues that affect clients as well as PingDirectory Server.

- *Log messages*
- *manage-certificates check-certificate-usability*
- *ldapsearch*
- *Using low-level TLS debugging*

Log messages

The following describes how to use the server's log messages to troubleshoot TLS-related issues.

To troubleshoot TLS-related issues, start by checking the server's access log. If the client can establish a TCP connection to the server, which must occur before TLS negotiation can start, the access log shows a `CONNECT` message with the following information:

- Source and destination address and port for the connection
- Protocol
- Selected client connection policy

The `CONNECT` message does not appear

If the `CONNECT` does not appear, the client might be unable to communicate with the server. The culprit can be a network problem, a firewall that is blocking attempts to communicate, or the client is trying to use an incorrect address or port.

The `CONNECT` message does appear

If the `CONNECT` message appears in the access log, it typically includes a `conn` element that specifies the connection ID. To view additional log messages for the client connection, use the **`search-logs`** tool. For example, if the connection ID is `12345`, the following command displays the complete set of associated log messages.

```
$ bin/search-logs --logFile logs/access conn=12345
```

If you are using LDAPS

If you are attempting to use LDAPS, one of the following log messages appears next:

- `SECURITY-NEGOTIATION` message – Indicates that the client and server successfully completed the negotiation process and that the issue likely occurred after the TLS session was established. This message also includes details about the negotiation, including the TLS protocol and the selected cipher suite.
- `DISCONNECT` message – The issue might involve a failure in the TLS-negotiation process. In such scenarios, the message usually includes a `reason` element that provides additional information about the reason for the disconnect.

  If the failure occurred during TLS negotiation, the usefulness of the `DISCONNECT` message depends in part on whether the failure occurred on the client or the server. For example, if the server decided to abort the negotiation, the message ideally contains the specific reason. If the problem occurred on the client, the log message likely contains only the general category for the failure.

> ⓘ **Note:**
>
> The TLS protocol does not provide a mechanism for conveying detailed error messages. Instead, it offers only a basic alert mechanism with a fixed set of alert types. For example, if a client does not trust the certificate chain that the server presents to it, the server might receive a generic alert like `certificate_unknown`, even if the client knows the precise reason for rejecting the chain. In such

instances, you might need to determine whether the client can provide additional details about the issue.

If the access log does not provide useful information

If the access log does not provide useful information, check the server error log. Although the error log does not normally include information about issues that relate to client communication, it provides helpful information in certain circumstances, like when an internal error within the server interferes with communication attempts.

manage-certificates check-certificate-usability
The **`manage-certificates`** tool offers a **`check-certificate-usability`** subcommand to examine a specified entry in a key store and to identify potential issues that might interfere with secure communication.

The **`check-certificate-usability`** tool completes the following tasks:

- Ensures that a specified entry in the key store includes a private key and a complete certificate chain
- Checks whether the certificate at the root of the chain is found in the Java virtual machine's (JVM's) default set of trusted certificates
- Ensures that the current time lies is within the validity window for all certificates in the chain
- Validates the signatures for all certificates in the chain
- Warns if the end-entity certificate is self-signed
- Warns if the end-entity certificate does not contain an extended key usage extension with the `serverAuth` usage
- Warns if the issuer certificates do not have a key usage extension with the `keyCertSign` usage
- Warns if the issuer certificates do not have a basic constraints extension indicating that it can operate as a certification authority

  If the chain violates a path length constraint, the **`check-certificate-usability`** tool reports an error.
- Ensures that the signature algorithm uses a strong message digest algorithm, like SHA-256

  The **`check-certificate-usability`** tool reports an error for weak digest algorithms like MD5 or SHA-1, and reports a warning for unrecognized digest algorithms.
- Ensures that none of the certificates that use an RSA key pair have a key size less than 2048 bits

The following example demonstrates the usage for the **`manage-certificates check-certificate-usability`** command and its output when no problems are identified.

```
$ bin/manage-certificates check-certificate-usability \
     --keystore config/keystore \
     --keystore-password-file config/keystore.pin \
     --alias server-cert

Successfully retrieved the certificate chain for alias 'server-cert':

Subject DN:  CN=ds1.example.com,O=Example Corp,C=US
Issuer DN:  CN=Example Intermediate CA,O=Example Corp,C=US
Validity Start Time: Tuesday, November 12, 2019 at 03:52:44 PM CST
                     (5 minutes, 45 seconds ago)
Validity End Time: Wednesday, November 11, 2020 at 03:52:44 PM CST
                   (364 days, 23 hours, 54 minutes, 14 seconds from now)
Validity State:  The certificate is currently within the validity window.
Signature Algorithm:  SHA-256 with RSA
Public Key Algorithm:  RSA (2048-bit)
SHA-1 Fingerprint:
 84:e4:00:b9:f0:6b:58:bb:ac:67:79:28:2f:43:9f:e3:ac:24:ee:98
SHA-256 Fingerprint:
 63:85:4d:2c:50:ea:a8:84:54:e0:73:9a:e7:5b:e7:1b:06:85:0e:
```

```
                    28:2b:76:a9:8b:57:fc:27:f7:60:81:48:41

Subject DN:  CN=Example Intermediate CA,O=Example Corp,C=US
Issuer DN:  CN=Example Root CA,O=Example Corp,C=US
Validity Start Time: Tuesday, November 12, 2019 at 03:52:42 PM CST
                     (5 minutes, 47 seconds ago)
Validity End Time: Monday, November 7, 2039 at 03:52:42 PM CST
                   (7299 days, 23 hours, 54 minutes, 12 seconds from now)
Validity State:  The certificate is currently within the validity window.
Signature Algorithm:  SHA-256 with RSA
Public Key Algorithm:  RSA (4096-bit)
SHA-1 Fingerprint:
 de:da:3d:fc:d4:1f:67:79:0a:a1:5a:cd:ca:4a:7e:a5:d3:46:88:27
SHA-256 Fingerprint:

 02:3c:af:ad:b7:07:81:89:45:48:d0:09:31:a8:90:c4:17:11:1c:00:11:fd:49:b2:2c:
   ba:ac:dd:c4:9f:03:36

Subject DN:  CN=Example Root CA,O=Example Corp,C=US
Issuer DN:  CN=Example Root CA,O=Example Corp,C=US
Validity Start Time: Tuesday, November 12, 2019 at 03:52:38 PM CST
                     (5 minutes, 51 seconds ago)
Validity End Time: Monday, November 7, 2039 at 03:52:38 PM CST
                   (7299 days, 23 hours, 54 minutes, 8 seconds from now)
Validity State:  The certificate is currently within the validity window.
Signature Algorithm:  SHA-256 with RSA
Public Key Algorithm:  RSA (4096-bit)
SHA-1 Fingerprint:
 8e:03:e4:58:e6:e3:59:9a:55:77:c0:88:3c:fa:d7:29:f4:ff:de:6c
SHA-256 Fingerprint:
 95:54:0d:e2:aa:48:29:c1:25:7c:20:69:c0:27:33:31:81:07:02:
                    2e:00:24:ae:49:5e:98:bd:a3:72:a5:05:26

OK:  The certificate chain is complete.  Each subsequent certificate is
the issuer for the previous certificate in the chain, and the chain ends
with a self-signed certificate.

OK:  Certificate 'CN=ds1.example.com,O=Example Corp,C=US' has a valid
signature.

OK:  Certificate 'CN=Example Intermediate CA,O=Example Corp,C=US' has a
valid signature.

OK:  Certificate 'CN=Example Root CA,O=Example Corp,C=US' has a valid
signature.

OK:  Certificate 'CN=ds1.example.com,O=Example Corp,C=US' will expire at
Wednesday, November 11, 2020 at 03:52:44 PM CST (364 days, 23 hours, 54
minutes, 14 seconds from now), which is not in the near future.

OK:  Issuer certificate 'CN=Example Intermediate CA,O=Example Corp,C=US'
will expire at Monday, November 7, 2039 at 03:52:42 PM CST (7299 days, 23
hours, 54 minutes, 12 seconds from now), which is not in the near future.

OK:  Issuer certificate 'CN=Example Root CA,O=Example Corp,C=US' will
expire at Monday, November 7, 2039 at 03:52:38 PM CST (7299 days, 23
hours, 54 minutes, 8 seconds from now), which is not in the near future.

OK:  Certificate 'CN=ds1.example.com,O=Example Corp,C=US' at the head of
the chain includes an extended key usage extension, and that extension
includes the serverAuth usage.

OK:  Issuer certificate 'CN=Example Intermediate CA,O=Example Corp,C=US'
includes a basic constraints extension, and the certificate chain
```

```
satisfies those constraints.

OK:  Issuer certificate 'CN=Example Intermediate CA,O=Example Corp,C=US'
includes a key usage extension with the keyCertSign usage flag set to
true.

OK:  Issuer certificate 'CN=Example Root CA,O=Example Corp,C=US' includes
a basic constraints extension, and the certificate chain satisfies those
constraints.

OK:  Issuer certificate 'CN=Example Root CA,O=Example Corp,C=US' includes
a key usage extension with the keyCertSign usage flag set to true.

OK:  Certificate 'CN=ds1.example.com,O=Example Corp,C=US' uses a signature
algorithm of 'SHA-256 with RSA', which is is considered strong.

OK:  Certificate 'CN=Example Intermediate CA,O=Example Corp,C=US' uses a
signature algorithm of 'SHA-256 with RSA', which is is considered strong.

OK:  Certificate 'CN=Example Root CA,O=Example Corp,C=US' uses a signature
algorithm of 'SHA-256 with RSA', which is is considered strong.

OK:  Certificate 'CN=ds1.example.com,O=Example Corp,C=US' has a 2048-bit
RSA public key, which is considered strong.

OK:  Certificate 'CN=Example Intermediate CA,O=Example Corp,C=US' has a
4096-bit RSA public key, which is considered strong.

OK:  Certificate 'CN=Example Root CA,O=Example Corp,C=US' has a 4096-bit
RSA public key, which is considered strong.

No usability errors or warnings were identified while validating the
certificate chain.
```

If any usability issues are identified, they might be responsible for communication problems.

ldapsearch

The **`ldapsearch`** command-line utility is a powerful tool for issuing searches against an LDAP directory server. It also provides a convenient method for troubleshooting a variety of issues, including problems that are relevant to TLS communication.

The following table details arguments that are the most useful for TLS-related communication.

TLS-related communication arguments and their descriptions

Argument	Description
--hostname {address}	Address of the server to which the connection is established
--port {port}	TCP port of the server to which the connection is established. The standard port for non-secure LDAP, or LDAP to be secured with StartTLS, is 389, and the standard port for secure LDAPS is 636. Many deployments use alternate ports, especially non-privileged ports above 1024.
--useSSL	The tool establishes an initially insecure LDAP connection, which is secured later with the StartTLS extended operation.

Argument	Description
--trustStorePath {path}	Path to the trust store that is used when determining whether to trust the certificate chain that the server presents during TLS negotiation. If neither this argument nor the --trustAll argument is provided, the tool prompts you interactively whether to trust server certificates that are not signed by an issuer in the Java virtual machine's (JVM's) default trust store.
--trustStoreFormat {format}	Format for the trust store, which is typically JKS or PKCS12.
--trustStorePassword {password}	Password that is required to access the contents of the trust store.
--trustStorePasswordFile {path}	Path to the file that contains the password that is required to access the contents of the trust store.
--trustAll	The tool blindly trusts all TLS certificate chains that are presented to it. Although this argument can prove useful for troubleshooting purposes, it is not recommended for general use.
--keyStorePath {path}	Path to the key store to use if a client certificate chain is presented to the server. ⓘ **Note:** Use this argument only when one of the following conditions is satisfied: ▪ The server is configured to require clients to present a certificate. ▪ You intend to use the certificate to authenticate through SASL EXTERNAL.
--keyStoreFormat {format}	Format for the key store, which is typically JKS or PKCS12.
--keyStorePassword {password}	Password to access the key store.
--keyStorePasswordFile {path}	Path to the file that contains the password necessary to access the key store.
--certNickname {alias}	Alias of the private key entry in the key store. Use when obtaining the certificate chain to present to the server.
--useSASLExternal	The client authenticates with the EXTERNAL SASL mechanism, which typically identifies the client using the certificate chain that is presented during TLS negotiation.

Argument	Description
--enableSSLDebugging	The tool activates the low-level TLS-debugging feature that the JVM provides.

The following command provides an example of the simplest method for testing TLS communication with PingDirectory Server.

```
$ bin/ldapsearch \
     --hostname ds1.example.com \
     --port 636 \
     --useSSL \
     --baseDN "" \
     --scope base \
     "(objectClass=*)"
The server presented the following certificate chain:

     Subject: CN=ds1.example.com,O=Example Corp,C=US
     Valid From: Tuesday, November 12, 2019 at 08:28:08 PM CST
     Valid Until: Wednesday, November 11, 2020 at 08:28:08 PM
 CST
     SHA-1 Fingerprint:

 6a:22:2a:bd:0b:1b:09:35:63:bc:12:3e:2c:9e:e7:70:bc:a4:73:de
     256-bit SHA-2 Fingerprint:

 7a:8c:e4:76:d4:47:15:fd:65:f5:26:0e:d2:55:77:d7:03:7a:e6:79:9f:bc:
        ae:93:2c:76:9c:01:fc:ef:15:38
     -
     Issuer 1 Subject: CN=Example Intermediate CA,O=Example
 Corp,C=US
     Valid From: Tuesday, November 12, 2019 at 08:28:06 PM CST
     Valid Until: Monday, November 7, 2039 at 08:28:06 PM CST
     SHA-1 Fingerprint:
 01:b3:70:8b:6c:11:43:87:3b:e9:bb:73:27:99:ea:fd:08:c4:db:ec
     256-bit SHA-2 Fingerprint:
 49:60:69:df:33:9d:26:d0:66:c9:6d:7b:0b:cb:3b:96:

 40:22:dc:6d:11:32:b7:c0:30:47:d6:7c:6a:19:cd:60
     -
     Issuer 2 Subject: CN=Example Root CA,O=Example Corp,C=US
     Valid From: Tuesday, November 12, 2019 at 08:28:03 PM CST
     Valid Until: Monday, November 7, 2039 at 08:28:03 PM CST
     SHA-1 Fingerprint:
 b4:83:55:db:82:e4:63:5c:3a:44:13:8f:88:44:e3:60:f2:53:80:48
     256-bit SHA-2 Fingerprint:

 e8:af:6f:ed:b9:0e:df:94:9c:20:29:53:a9:74:44:a9:17:b4:08:65:c8:19:c1:fb:
        34:34:a1:90:83:8a:d5:12

Do you wish to trust this certificate?  Enter 'y' or 'n': y
dn:
objectClass: top
objectClass: ds-root-dse
startupUUID: 8d574122-4584-4522-96d9-0cdcb9d2e339
startTime: 201911130611492

# Result Code:  0 (success)
# Number of Entries Returned:  1
```

Trust stores and trust-related arguments

If no trust-related arguments are provided, the tool uses the JVM's default trust store to verify whether to trust the certificate chain, based on the information that it contains. If a trusted authority has signed the server certificate, the negotiation process continues without further interaction.

If the chain cannot be trusted, based on the information in the JVM-default trust store, **`ldapsearch`** prompts you interactively about whether to trust the certificate. If you accept the chain, the client and server complete the negotiation process, and the client sends the search request to the server. If the search succeeds, the server can communicate over TLS.

To test with a trust store instead of being prompted interactively, use the --trustStorePath argument that points to the appropriate trust store. If you are using a Java Keystore (JKS) trust store, you might not need to provide the trust store password. If you are using a PKCS #12 trust store, you need to provide the trust store password. The following code provides an example.

```
$ bin/ldapsearch \
     --hostname ds1.example.com \
     --port 636 \
     --useSSL \
     --trustStorePath config/truststore.p12 \
     --trustStorePasswordFile config/truststore.pin \
     --trustStoreFormat PKCS12 \
     --baseDN "" \
     --scope base \
     "(objectClass=*)"
dn:
objectClass: top
objectClass: ds-root-dse
startupUUID: c8724159-8c37-45eb-b210-879bfcf74ad6
startTime: 20191113154023Z

# Result Code:  0 (success)
# Number of Entries Returned:  1
```

Client certificate chains and key stores

To present a client certificate chain to the server, either because the server's connection handler is configured with an ssl-client-auth-policy value of required or because you plan to use the certificate to authenticate by way of the SASL EXTERNAL mechanism, provide at least the key store and its corresponding password. You can also specify the alias of the certificate chain to present, which is recommended if your client key store contains multiple certificates. The following code provides an example.

```
$ bin/ldapsearch \
     --hostname ds1.example.com \
     --port 636 \
     --useSSL \
     --trustStorePath config/truststore.p12 \
     --trustStorePasswordFile config/truststore.pin \
     --trustStoreFormat PKCS12 \
     --keyStorePath client-keystore \
     --keyStorePasswordFile client-keystore.pin \
     --certNickname client-cert \
     --useSASLExternal \
     --baseDN "" \
     --scope base \
     "(objectClass=*)"
```

```
dn:
objectClass: top
objectClass: ds-root-dse
startupUUID: c8724159-8c37-45eb-b210-879bfcf74ad6
startTime: 20191113154023Z

# Result Code:  0 (success)
# Number of Entries Returned:  1
```

If you need to further troubleshoot a TLS-related issue

If you encounter a TLS-related issue that you cannot resolve by examining the **`ldapsearch`** output or the server logs, use the --enableSSLDebugging option to enable the JVM's support for low-level debugging of TLS processing. For more information, see *Using low-level TLS debugging* on page 360.

Using low-level TLS debugging

Use tools other than the command-line tools that are provided with PingDirectory Server for performing low-level TLS debugging.

Before you begin

> ⓘ **Note:**
>
> If you need to use low-level debugging options, enable the Java Virtual Machine (JVM)'s support for TLS debugging. Many of the command-line tools that are provided with PingDirectory Server, such as **`ldapsearch`**, offer an --enableSSLDebugging argument that simplifies this process.

Steps

1. In the `config/java.properties` file, add the following line to the set of properties for the appropriate tool.

   ```
   -Djavax.net.debug=all
   ```

2. For the changes to take effect, run the **`bin/dsjavaproperties`** command.

Next steps

The next time the tool is run, an output is generated detailing the TLS-related processing that the JVM is performing. You and the Ping Identity support team can use the output to identify the issue.

Domain Name Service (DNS) caching

If needed, two global configuration properties can be used to control the caching of host name-to-numeric IP address (DNS lookup) results returned from the name resolution services of the underlying operating system. Use the `dsconfig` tool to configure these properties.

network-address-cache-ttl

Sets the Java system property `networkaddress.cache.ttl`, and controls the length of time in seconds that a host name-to-IP address mapping can be cached. The default behavior is to keep resolution results for one hour (3600 seconds). This setting applies to the server and all extensions loaded by the server.

network-address-outage-cache-enabled

Caches host name-to-IP address results in the event of a DNS outage. This is set to `true` by default, meaning name resolution results are cached. Unexpected service interruptions may occur

during planned or unplanned maintenance, network outages or an infrastructure attack. This cache may allow the server to function during a DNS outage with minimal impact. This cache is not available to server extensions.

To reduce delays due to unnecessary DNS lookups, follow these recommendations:

- Maintain a connection pool in the client app rather than opening new connections for each bind.
- Add appropriate records to DNS, including PTR records.
- Add `options timeout:1` and/or `options single-request` in `/etc/resolv.conf`.
- If IPv6 requests specifically are causing issues, Add `-Djava.net.preferIPv4Stack=true` to the `start-server.java-args` line in PingDirectory's `config/java.properties` so that running `bin/dsjavaproperties` and restarting the server will no longer issue IPv6 PTR requests.

IP address reverse name lookups

Ping Identity servers perform some numeric IP address-to-host name lookups, including the following:

- Binding to the Directory: Decoding, examining, or evaluating a DNS bind rule
- Logging: Logging information to certain monitors or writing to the error log
- JMX: Creating a server socket
- Key Management: Generating a truststore
- Replication Server: Creating an SSL socket
- Replication Session Management: Obtaining a session or performing a handshake with a replication server
- SASL Authentication: Applying configuration changes
- SMTP Alert Handler: Initializing or sending an alert notification

Address masks configured in Access Control Lists (ACIs), Connection Handlers, Connection Criteria, and Certificate handshake processing may trigger implicit reverse name lookups. For more information about how address masks are configured in the server, review the following information for each server:

- ACI dns: bind rules under *Managing Access Control* (PingDirectory Server and PingDirectoryProxy Servers)
- `ds-auth-allowed-address`: *Adding Operational Attributes that Restrict Authentication* (PingDirectory Server)
- Connection Criteria: *Restricting Server Access Based on Client IP Address* (PingDirectory Server and PingDirectoryProxy Servers)
- Connection Handlers: *Restrict Server Access Using Connection Handlers* (Configuration Reference Guide for all PingData servers)

Configure the synchronization environment with dsconfig

The `dsconfig` tool can be used to configure any part of PingDataSync Server, but will likely be used for more fine-grained adjustments. If configuring a Sync Pipe for the first time, use the `bin/create-sync-pipe-config` tool to guide through the necessary Sync Pipes creation steps.

Configure server groups with dsconfig interactive

In a typical deployment, one PingDataSync Server and one or more redundant failover servers are configured. Primary and secondary servers must have the same configuration settings to ensure proper operation. To enable this, assign all servers to a server group using the `dsconfig` tool. Any change to one server will automatically be applied to the other servers in the group.

Run the `dsconfig` command and set the global configuration property for server groups to `all-servers`. On the primary PingDataSync Server, perform the following steps:

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

Updates to servers in the group are made using the `--applyChangeToservergroup` option of the `dsconfig` command. To apply the change to one server in the group, use the `--applyChangeTo single-server` option. If additional servers are added to the topology, the `setup` tool will copy the configuration from the primary server to the new server(s).

Start the Global Sync configuration with dsconfig interactive

About this task

After the Synchronization topology is configured, perform the following steps to start the Global Sync configuration, which will use only those Sync Pipes that have been started:

Steps

1. On the `dsconfig` main menu, type the number corresponding to the Global Sync Configuration.
2. On the Global Sync Configuration menu, type the number corresponding to view and edit the configuration.
3. On the GlobalSync Configuration Properties menu, type the number corresponding to setting the started property, and then follow the prompts to set the value to `TRUE`.
4. On the GlobalSync Configuration Properties menu, type `f` to save and apply the changes.

Prepare external server communication

About this task

The `prepare-endpoint-server` tool sets up any communication variances that may occur between PingDataSync Server and the external servers. Typically, directory servers can have different security settings, privileges, and passwords configured on the Sync Source that might reject the import of entries in the Sync Destination.

The `prepare-endpoint-server` tool also creates a Sync User Account and its privileges on all of the external servers (see *Sync user account* on page 1309 for more detailed information). The `prepare-endpoint-server` tool verifies that the account has the proper privileges to access the `firstChangeNumber` and `lastChangeNumber` attributes in the root DSE entry so that it can access the latest changes. If the Sync User does not have the proper privileges, PingDataSync Server displays a warning message, which is saved in the `logs/prepare- endpoint-server.log` file.

> ⓘ **Note:**
>
> If the synchronization topology was created using the `create-sync-pipe-config` tool, this command does not need to be run. It is already part of the `create-sync-pipe-config` process.

Perform the following steps to prepare PingDataSync Server for external server communication:

Steps

1. Use the `prepare-endpoint-server` tool to prepare the directory server instances on the remote host for synchronization as a data source for the subtree, `dc=example,dc=com`. If the user account is not present on the external server, it will be created if a parent entry exists.

```
$ bin/prepare-endpoint-server \
  --hostname sun-ds1.example.com \
  --port 21389 \
  --syncServerBindDN "cn=Sync User,dc=example,dc=com" \
  --syncServerBindPassword secret \
  --baseDN "dc=example,dc=com" \
  --isSource
```

2. When prompted,enter the bind DN and password to create the user account. This step enables the change log database and sets the `changelog-maximum-age` property.
3. Repeat steps 1–2 for any other external source servers.
4. For the destination servers, repeat steps 2–3 and include the `--isDestination` option. If destination servers do not have any entries, a "Denied" message will display when creating the `cn=Sync User` entry.

```
$ bin/prepare-endpoint-server \
  --hostname PingIdentity-ds1.example.com \
  --port 33389 \
  --syncServerBindDN "cn=Sync User,cn=Root DNs,cn=config" \
  --syncServerBindPassword sync \
  --baseDN "dc=example,dc=com" \
  --isDestination
```

5. Repeat step 4 for any other destination servers.

HTTP connection handlers

HTTP Connection Handlers are responsible for managing the communication with HTTP clients and invoking servlets to process requests from those clients. They can also be used to host web applications on the server. Each HTTP connection handler must be configured with one or more HTTP servlet extensions and zero or more HTTP operation log publishers.

If the HTTP Connection Handler cannot be started (for example, if its associated HTTP Servlet Extension fails to initialize), then this will not prevent the entire Directory Proxy Server from starting. The server's `start-server` tool will output any errors to the error log. This allows the server to continue serving LDAP requests even with a bad servlet extension.

The configuration properties available for use with a HTTP connection handler include:

listen-address

Specifies the address on which the connection handler will listen for requests from clients. If not specified, then requests will be accepted on all addresses bound to the system.

listen-port

Specifies the port on which the connection handler will listen for requests from clients. Required.

use-ssl

Indicates whether the connection handler will use SSL/TLS to secure communications with clients (whether it uses HTTPS rather than HTTP). If SSL is enabled, then `key-manager-provider` and `trust-manager-provider` values must also be specified.

http-servlet-extension

Specifies the set of servlet extensions that will be enabled for use with the connection handler. You can have multiple HTTP connection handlers (listening on different address/port combinations) with identical or different sets of servlet extensions. At least one servlet extension must be configured.

http-operation-log-publisher

Specifies the set of HTTP operation log publishers that should be used with the connection handler. By default, no HTTP operation log publishers will be used.

key-manager-provider

Specifies the key manager provider that will be used to obtain the certificate presented to clients if SSL is enabled.

trust-manager-provider

Specifies the trust manager provider that will be used to determine whether to accept any client certificates presented to the server.

num-request-handlers

Specifies the number of threads that should be used to process requests from HTTP clients. These threads are separate from the worker threads used to process other kinds of requests. The default value of zero means the number of threads will be automatically selected based on the number of CPUs available to the JVM.

web-application-extension

Specifies the Web applications to be hosted by the server.

Configure an HTTP connection handler

About this task

A HTTP connection handler has two dependent configuration objects: one or more HTTP servlet extensions and optionally, a HTTP log publisher. The HTTP servlet extension and log publisher must be configured prior to configuring the HTTP connection handler. The log publisher is optional but in most cases, you want to configure one or more logs to troubleshoot any issues with your HTTP connection.

Steps

1. The first step is to configure your HTTP servlet extensions. The following example uses the ExampleHTTPServletExtension in the Server SDK.

```
$ bin/dsconfig create-http-servlet-extension \
  --extension-name "Hello World Servlet" \
  --type third-party \
  --set
"extensionclass:com.unboundid.directory.sdk.examples.ExampleHTTPServletEx
tension" \
  --set "extension-argument:path=/" \
  --set "extension-argument:name=example-servlet"
```

2. Next, configure one or more HTTP log publishers. The following example configures two log publishers: one for common access; the other, detailed access. Both log publishers use the default configuration settings for log rotation and retention.

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

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

3. Configure the HTTP connection handler by specifying the HTTP servlet extension and log publishers. Note that some configuration properties can be later updated on the fly while others, like listen-port, require that the HTTP Connection Handler be disabled, then re-enabled for the change to take effect.

```
$ bin/dsconfig create-connection-handler \
  --handler-name "Hello World HTTP Connection Handler" \
  --type http \
  --set enabled:true \
  --set listen-port:8443 \
  --set use-ssl:true \
  --set "http-servlet-extension:Hello World Servlet" \
  --set "http-operation-log-publisher:HTTP Common Access Logger" \
  --set "http-operation-log-publisher:HTTP Detailed Access Logger" \
  --set "key-manager-provider:JKS" \
  --set "trust-manager-provider:JKS"
```

4. By default, the HTTP Connection Handler has an advanced monitor entry property, `keep- stats`, that is set to `TRUE` by default. You can monitor the connection handler using the `ldapsearch` tool.

```
$ bin/ldapsearch --baseDN "cn=monitor" \
  "(objectClass=ds-http-connection-handler-statistics-monitor-entry)"
```

HTTP correlation IDs

A typical request to a software system is handled by multiple subsystems, many of which may be distinct servers residing on distinct hosts and locations. Tracing the request flow on distributed systems can be challenging, as log messages are scattered across various systems and intermingled with messages for other requests. To make this easier, a correlation ID can be assigned to a request, which is then added to every associated operation as the request flows through the larger system. The correlation ID allows related log messages to be easily located and grouped. The server supports correlation IDs for all HTTP requests received through its HTTP(S) Connection Handler.

When a HTTP request is received, it is automatically assigned a correlation ID. This ID can be used to correlate HTTP responses with messages recorded to the HTTP Detailed Operation log and the trace log. For specific web APIs, the correlation ID may also be passed to the LDAP subsystem. For the SCIM 1, Delegated Admin, Consent, and Directory REST APIs, the correlation ID will also appear with associated requests in LDAP logs in the `correlationID` key. The correlation ID is also used as the default client request ID value in Intermediate Client Request Controls used by the SCIM 2, Consent, and Directory REST APIs. Values related to the Intermediate Client Request Control appear in the LDAP logs in the `via` key, and are forwarded to downstream LDAP servers when received by the PingDirectoryProxy server. The correlation ID header is also added to requests forwarded by the PingDataGovernance gateway.

For Server SDK extensions that have access to the current HttpServletRequest, the current correlation ID can be retrieved as a string through the `HttpServletRequest's com.pingidentity.pingdata.correlation_id` attribute. For example:

```
(String) request.getAttribute("com.pingidentity.pingdata.correlation_id");
```

Configure HTTP Correlation ID Support

Correlation ID support is enabled by default for each HTTP Connection Handler.

- To enable correlation ID support for the HTTPS Connection Handler:

```
$ bin/dsconfig set-connection-handler-prop \
                    --handler-name "HTTPS Connection Handler" \
                    --set use-correlation-id-header:true
```

- To disable correlation ID support for the HTTPS Connection Handler:

```
$ bin/dsconfig set-connection-handler-prop \
```

```
                    --handler-name "HTTPS Connection Handler" \
                    --set use-correlation-id-header:false
```

Configuring the correlation ID response header

- The server will generate a correlation ID for every HTTP request and send it in the response through the `Correlation-Id` response header. This response header name can be customized. The following example changes the `correlation-id-response-header` property to"X-Request-Id."

```
$ bin/dsconfig set-connection-handler-prop \
                    --handler-name "HTTPS Connection Handler" \
                    --set correlation-id-response-header:X-Request-Id
```

Accepting an incoming correlation ID from the request

- By default, the server generates a new, unique correlation ID for each HTTP request, and ignores any correlation ID that may be set on the request. This can be changed by designating the names of one or more HTTP request headers that contain an existing correlation ID value. This enables the server to integrate with a larger system consisting of every servers using correlation IDs.

```
$ bin/dsconfig set-connection-handler-prop \
                    --handler-name "HTTPS Connection Handler" \
                    --set correlation-id-request-header:X-Request-Id \
                    --set correlation-id-request-header:X-Correlation-Id \
                    --set correlation-id-request-header:Correlation-Id \
                    --set correlation-id-request-header:X-Amzn-Trace-Id
```

HTTP Correlation ID Example Use

In this example, a request to the Directory REST API is made and the correlation ID enables finding HTTP-specific log messages with LDAP-specific log messages. The response to the API call includes a `Correlation-Id` header with the value `a54aee33-c6c6-4467-be25-efd1db7a8b76`.

```
GET /directory/v1/me?includeAttributes=mail HTTP/1.1
                Accept: */*
                Accept-Encoding: gzip, deflate
                Authorization: Bearer ...
                Connection: keep-alive
                Host: localhost:1443
                User-Agent: HTTPie/0.9.9
                HTTP/1.1 200 OK
                Content-Length: 266
                Content-Type: application/hal+json
                Correlation-Id: ee919049-6710-4594-9c66-28b4ada4b127
                Date: Fri, 02 Nov 2018 15:16:50 GMT
                Request-Id: 369
                {
                "_dn": "uid=user.86,ou=People,dc=example,dc=com",
                "_links": {
                "schemas": [
                {
                "href": "https://localhost:1443/directory/v1/schemas/
                inetOrgPerson"
                }
                ],
                "self": {
                "href": "https://localhost:1443/directory/v1/
                uid=user.86,ou=People,dc=example,dc=com"
                }
                },
                "mail": [
```

```
                "user.86@example.com"
                ]
                }
```

This correlation ID can be used to search the HTTP trace log for matching log records, as follows:

```
$ grep 'correlationID="ee919049-6710-4594-9c66-28b4ada4b127"'
                        PingDirectory/logs/debug-trace
                        [02/Nov/2018:10:16:50.294 -0500] HTTP REQUEST
 requestID=369
                        correlationID="ee919049-6710-4594-9c66-28b4ada4b127"
 product="Ping Identity
                        Directory Server" instanceName="ds1"
 startupID="W9ikqA==" threadID=52358 from=
                        [0:0:0:0:0:0:0:1]:58918 method=GET
                        url="https://0:0:0:0:0:0:0:1:1443/directory/v1/me?
includeAttributes=mail"
                        [02/Nov/2018:10:16:50.526 -0500] DEBUG ACCESS-TOKEN-
VALIDATOR-PROCESSING
                        requestID=369
 correlationID="ee919049-6710-4594-9c66-28b4ada4b127"
                        msg="Identity Mapper with DN 'cn=User ID Identity
 Mapper,cn=Identity
                        Mappers,cn=config' mapped ID 'user.86' to entry DN
                        'uid=user.86,ou=people,dc=example,dc=com'"
                        [02/Nov/2018:10:16:50.526 -0500] DEBUG ACCESS-TOKEN-
VALIDATOR-PROCESSING
                        requestID=369
 correlationID="ee919049-6710-4594-9c66-28b4ada4b127"
                        accessTokenId="201811020831" msg="Token Validator
 'Mock Access Token
                        Validator' validated access token with active =
 'true', sub = 'user.86', owner
                        = 'uid=user.86,ou=people,dc=example,dc=com',
 clientId = 'client1', scopes =
                        'ds', expiration = 'none', not-used-before = 'none',
 current time = 'Nov 2,
                        2018 10:16:50 AM CDT' "
                        [02/Nov/2018:10:16:50.531 -0500] HTTP RESPONSE
 requestID=369
                        correlationID="ee919049-6710-4594-9c66-28b4ada4b127"
                        accessTokenId="201811020831" product="Ping Identity
 Directory Server"
                        instanceName="ds1" startupID="W9ikqA=="
 threadID=52358 statusCode=200
                        etime=236.932 responseContentLength=266
                        [02/Nov/2018:10:16:50.531 -0500] DEBUG HTTP-FULL-
REQUEST-AND-RESPONSE
                        requestID=369
 correlationID="ee919049-6710-4594-9c66-28b4ada4b127"
                        accessTokenId="201811020831" product="Ping Identity
 Directory Server"
                        instanceName="ds1" startupID="W9ikqA=="
 threadID=52358 from=
                        [0:0:0:0:0:0:0:1]:58918 method=GET
                        url="https://0:0:0:0:0:0:0:1:1443/directory/v1/me?
includeAttributes=mail"
                        statusCode=200 etime=236.932
 responseContentLength=266 msg="
```

The LDAP log messages associated with this request can also be located:

```
$ grep 'correlationID="ee919049-6710-4594-9c66-28b4ada4b127"'
```

```
                                PingDirectory/logs/access
                                [02/Nov/2018:10:16:50.529 -0500] SEARCH
 RESULT instanceName="ds1"
                                threadID=52358 conn=-371045 op=1657393
 msgID=1657394 origin="Directory REST
                                API" httpRequestID="369"
 correlationID="ee919049-6710-4594-9c66-28b4ada4b127"

 authDN="uid=user.86,ou=people,dc=example,dc=com" requesterIP="internal"

 requesterDN="uid=user.86,ou=People,dc=example,dc=com"
                                requestControls="1.3.6.1.4.1.30221.2.5.2"
 via="app='PingDirectoryds1'
                                clientIP='0:0:0:0:0:0:0:1'
 sessionID='201811020831' requestID='ee919049-6710-
                                4594-9c66-28b4ada4b127'"
 base="uid=user.86,ou=people,dc=example,dc=com"
                                scope=0 filter="(&)"
 attrs="mail,objectClass" resultCode=0
                                resultCodeName="Success" etime=0.684
 entriesReturned=1
                                [02/Nov/2018:10:16:50.530 -0500] EXTENDED
 RESULT instanceName="ds1"
                                threadID=52358 conn=-371046 op=1657394
 msgID=1657395 origin="Directory REST
                                API" httpRequestID="369"
 correlationID="ee919049-6710-4594-9c66-28b4ada4b127"
                                authDN="cn=Internal
 Client,cn=Internal,cn=Root DNs,cn=config"
                                requesterIP="internal"
 requesterDN="cn=Internal Client,cn=Internal,cn=Root
                                DNs,cn=config"
 requestControls="1.3.6.1.4.1.30221.2.5.2"
                                via="app='PingDirectory-ds1'
 clientIP='0:0:0:0:0:0:0:1'
                                sessionID='201811020831'
 requestID='ee919049-6710-4594-9c66-28b4ada4b127'"
                                requestOID="1.3.6.1.4.1.30221.1.6.1"
 requestType="Password Policy State"
                                resultCode=0 resultCodeName="Success"
 etime=0.542 usedPrivileges="bypassacl,password-reset"
 responseOID="1.3.6.1.4.1.30221.1.6.1"
                                responseType="Password Policy State"
                                dn="uid=user.86,ou=People,dc=example,dc=com"
                                [02/Nov/2018:10:16:50.530 -0500] SEARCH
 RESULT instanceName="ds1"
                                threadID=52358 conn=-371048 op=1657397
 msgID=1657398 origin="Directory REST
                                API" httpRequestID="369"
 correlationID="ee919049-6710-4594-9c66-28b4ada4b127"
                                authDN="cn=Internal
 Client,cn=Internal,cn=Root DNs,cn=config"
                                requesterIP="internal"
 requesterDN="cn=Internal Client,cn=Internal,cn=Root
                                DNs,cn=config"
 requestControls="1.3.6.1.4.1.30221.2.5.2"
                                via="app='PingDirectoryds1'
 clientIP='0:0:0:0:0:0:0:1'
                                sessionID='201811020831'
 requestID='ee919049-6710-4594-9c66-28b4ada4b127'"
                                base="cn=Default Password Policy,cn=Password
 Policies,cn=config" scope=0
                                filter="(&)" attrs="dscfg- password-
attribute" resultCode=0
```

```
                                    resultCodeName="Success" etime=0.065
 preAuthZUsedPrivileges="bypassacl,config-read" entriesReturned=1
```

Resync tool

The `resync` tool provides bulk synchronization that can be used to verify the synchronization setup. The tool operates on a single Sync Pipe at a time, retrieves entries from the Sync Source in bulk, and compares the source entries with the corresponding destination entries. If destination entries are missing or attributes are changed, they are updated.

The command provides a `--dry-run` option that can be used to test the matches between the Sync Source and Destination, without committing any changes to the target topology. The `resync` tool also provides options to write debugging output to a log.

> ⓘ **Note:**
>
> The `resync` tool should be used for relatively small datasets. For large deployments, export entries from the Sync Source into an LDIF file, run the `bin/translate-ldif` tool to translate and filter the entries into the destination format, and then import the result LDIF file into the Sync Destination.

Use the `resync--help` command for more information and examples. Logging is located in `logs/tools/resync.log` and `logs/tools/resync-errors.log`. If necessary, the logging location can be changed with the `--logFilePath` option.

Test attribute and DN maps

The `resync` tool can be used to test how attribute maps and DN maps are configured by synchronizing a single entry. If the `--logFilePath` and `--logLevel` options are specified, the `resync` tool generates a log file with details.

Use the`--dry-run` option and specify a single entry, such as `uid=user.0`. Any logging performed during the operation appears in `logs/tools/resync.log`.

```
$ bin/resync --pipe-name sun-to-ds-sync-pipe \
  --sourceSearchFilter "(uid=user.0)" \
  --dry-run \
  --logLevel debug
```

Verify the synchronization configuration

The most common use for the `resync` tool is to test that the Sync Pipe configuration has been set up correctly. For example, the following procedure assumes that the configuration was set up with the Sync Source topology (two replicated Sun Directory servers) with 2003 entries; the Sync Destination topology (two replicated PingData PingDirectory Server) has only the base entry and the `cn=Sync User` entry. Both source and destination topologies have their LDAP Change Logs enabled. Because both topologies are not actively being updated, the `resync` tool can be run with one pass through the entries.

Use`resync` with the `--dry-run` option to check the synchronization configuration. The output displays a timestamp that can be tracked in the logs.

```
$ bin/resync --pipe-name sun-to-ds-sync-pipe \
  --numPasses 1 \
  --dry-run
Starting Pass 1
Status after completing all passes[20/Mar/2010:10:20:07 -0500]
---------------------------------
Source entries retrieved 2003
Entries missing 2002
Entries out-of-sync 1
Duration (seconds) 4
```

```
Resync completed in 4 s.
0 entries were in-sync, 0 entries were modified, 0 entries were created,
1 entries are still out-of-sync, 2002 entries are still missing, and
0 entries could not be processed due to an error
```

Populate an empty sync destination topology

About this task

The following procedure uses the `resync` tool to populate an empty Sync Destination topology for small datasets. For large deployments, use the `bin/translate-ldif`.

In this example, assume that the Sync Destination topology has only the base entry (`dc=example,dc=com`) and the `cn=Sync User` entry. Perform the following steps to populate an empty Sync Destination:

Steps

1. Run the `resync` command with the log file path and with the log level `debug`. Logging is located in `logs/tools/resync.log` and `logs/tools/resync-errors.log`.

   ```
   $ bin/resync --pipe-name sun-to-ds-sync-pipe \
     --numPasses 1 \
     --logLevel debug
   ```

2. Open the `logs/resync-failed-DNs.log` file in a text editor to locate the error and fix it. An entry cannot be created because the parent entry does not exist.

   ```
   # Entry '(see below)' was dropped because there was a failure at the
   resource:
   Failed to create entry uid=mlott,ou=People,dc=example,dc=com. Cause:
   LDAPException(resultCode=no such object, errorMessage='Entry
   uid=user.38,ou=People,dc=example,dc=com cannot be added because its parent
   entry ou=People,dc=example,dc=com does not exist in the server',
   matchedDN='dc=example,dc=com')
   (id=1893859385ResourceOperationFailedException.java:126 Build
   revision=4881)
   dn: uid=user.38,ou=People,dc=example,dc=com
   ```

3. Rerun the `resync` command. The command creates the entries in the Sync Destination topology that are present in the Sync Source topology.

   ```
   $ bin/resync --pipe-name sun-to-ds-sync-pipe
   ```

   ```
   ...(output from each pass)...
   Status after completing all passes[20/Mar/2016:14:23:33 -0500]
   ----------------------------
   Source entries retrieved 160
   Entries in-sync 156
   Entries created 4
   Duration (seconds) 11

   Resync completed in 12 s.

   156 entries were in-sync, 0 entries were modified, 4 entries were created,
   0 entries are still out-of-sync, 0 entries are still missing, and 0
   ```

```
entries could not be processed due to an error
```

> ⓘ **Note:**
>
> If importing a large amount of data into a PingData PingDirectory Server, run `export-ldif` and `import-ldif` on the newly populated backend for most efficient disk space use. If needed, run `dsreplication initialize` to propagate the efficient layout to additional replicas.

Set the synchronization rate

About this task

The `resync` command has a `--ratePerSecondFile` option that enables a specific synchronization rate. The option can be used to adjust the rate during off-peak hours, or adjust the rate based on measured loads for very long operations.

> ⓘ **Note:**
>
> The `resync` command also has a `--ratePerSecond` option. If this option is not provided, the tool operates at the maximum rate.

Run the `resync` tool first at 100 operations per second, measure the impact on the source servers, then adjust as desired. The file must contain a single positive integer number surrounded by white space. If the file is updated with an invalid number, the rate is not updated.

Steps

1. Create a text file containing the rate. The number must be a positive integer surrounded by white space.

```
$ echo ’100 ’ >rate.txt
```

2. Run the `resync` command with the `--ratePerSecondFile` option.

```
$ bin/resync --pipe-name "sun-to-ds-sync-pipe" \
  --ratePerSecondPath rate.txt
```

Synchronize a specific list of DNs

About this task

The `resync` command enables synchronizing a specific set of DNs that are read from a file using the `--sourceInputFile` option. This option is useful for large datasets that require faster processing by targeting individual base-level searches for each source DN in the file. If any DN fails (parsing, search, or process errors), the command creates an output file of the skipped entries (`resync-failed-DNs.log`), which can be run again.

The file must contain only a list of DNs in LDIF format with `dn:` or `dn::`. The file can include comment lines. All DNs can be wrapped and are assumed to be wrapped on any lines that begin with a space followed by text. Empty lines are ignored.

Small files can be created manually. For large files, use `ldapsearch` to create an LDIF file, as follows:

Steps

1. Run an `ldapsearch` command using the special OID "1.1" extension, which only returns the DNs in the DIT. For example, on the Sync Source directory server, run the following command:

```
$ bin/ldapsearch --port 1389 \
  --bindDN "uid=admin,dc=example,dc=com \
  --baseDN dc=example,dc=com \
  --searchScope sub "(objectclass=*)" "1.1" > dn.ldif
```

2. Run the `resync` command with the file.

```
$ bin/resync --pipe-name "sun-to-ds-pipe" \
  --sourceInputFile dn.ldif
```

```
Starting pass 1
[20/Mar/2016:10:32:11 -0500]
-------------------------------------
Resync pass 1
Source entries retrieved 1999
Entries created 981
Current pass, entries processed 981
Duration (seconds) 10
Average ops/second 98
Status after completing all passes[20/Mar/2016:10:32:18 -0500]
---------------------------------
Source entries retrieved 2003
Entries created 2003
Duration (seconds) 16
Average ops/second 98
Resync completed in 16 s.
0 entries were in-sync, 0 entries were modified, 2003 entries were
created, 0 entries are still out-of-sync, 0 entries are still missing, and
0 entries could not be processed due to an error
```

3. View the `logs/tools/resync-failed-DNs.log` to determine skipped DNs. Correct the source DNs file, and rerun the `resync` command.

Realtime-sync tool

The `bin/realtime-sync` tool controls starting and stopping synchronization globally or for individual Sync Pipes. The tool also provides features to set a specific starting point for real-time synchronization.

To see the current status of real-time synchronization, view the monitor properties: `num-sync-ops-in-flight`, `num-ops-in-queue`, and `source-unretrieved-changes`. For example, use `ldapsearch`to view a specific Sync Pipe's monitor information:

```
$ bin/ldapsearch --baseDN cn=monitor \
--searchScope sub "(cn=Sync Pipe Monitor: PIPE_NAME)"
```

The Stats Logger can also be used to view status. See the *PingIdentityPingDirectory Server Administration Guide* for details.

Start real-time synchronization globally

About this task

The `realtime-sync` tool assumes that the synchronization topology is configured correctly.

Perform the following steps to start real time synchronization globally:

Steps

1. Run the tool from the `<server-root>/bin` directory. This example assumes that a single Sync Pipe called "dsee-to-ds-sync-pipe" exists.

```
$ bin/realtime-sync start --pipe-name "dsee-to-ds-sync-pipe" \
  --port 389 \
  --bindDN "uid=admin,dc=example,dc=com" \
  --bindPassword secret
```

2. If more than one Sync Pipe is configured, specify each using the `--pipe-name` option. The following example starts synchronization for a bidirectional synchronization topology.

```
$ bin/realtime-sync start --pipe-name "Sun DS to DS" \
  --pipe-name "DS to Sun DS" \
  --port 389 \
  --bindDN "uid=admin,dc=example,dc=com" \
  --bindPassword secret
```

Start or Pause synchronization

Pause or start synchronization by using the `start` and `stop` subcommands. If synchronization is stopped and then restarted, changes made at the Sync Source while synchronization was stopped will still be detected and applied. Synchronization for individual Sync Pipes can be started or stopped using the `--pipe-name` argument. If the `--pipe-name` argument is omitted, then synchronization is started or stopped globally.

The following command stops all Sync Pipes:

```
$ bin/realtime-sync stop --port 389 \
  --bindDN "uid=admin,dc=example,dc=com" \
  --bindPassword secret \
  --no-prompt
```

If a topology has two Sync Pipes, Sync Pipe1 and Sync Pipe2, the following command stops Sync Pipe1.

```
$ bin/realtime-sync stop --pipe-name "Sync Pipe1" \
  --port 389 \
  --bindDN "uid=admin,dc=example,dc=com" \
  --bindPassword secret --no-prompt
```

Set startpoints

About this task

Startpoints instruct the Sync Pipe to ignore all changes made prior to the current time. Once synchronization is started, only changes made after this command is run will be detected at the Sync Source and applied at the Sync Destination.

The `set-startpoint` subcommand is often run during the initial setup prior to starting real-time synchronization. It should be run prior to initializing the data in the Sync Destination.

The `set-startpoint` subcommand can start synchronization at a specific change log number, or back at a state that occurred at a specific time. For example, synchronization can start 10 minutes prior to the current time.

Perform the following steps to set a synchronization startpoint:

Steps

1. If started, stop the synchronization topology globally with the following command:

```
$ bin/realtime-sync stop --pipe-name "Sync Pipe1" \
  --port 389 \
  --bindDN "uid=admin,dc=example,dc=com" \
  --bindPassword secret \
  --no-prompt
```

2. Set the startpoint for the synchronization topology. Any changes made before setting this command will be ignored.

```
$ bin/realtime-sync set-startpoint --pipe-name "Sync Pipe1" \
  --port 389 \
  --bindDN "uid=admin,dc=example,dc=com" \
  --bindPassword secret \

  --no-prompt \
  --beginning-of-changelog
```

```
Set StartPoint task 2011072109564107 scheduled to start immediately
[21/Jul/2016:09:56:41 -0500] severity="INFORMATION" msgCount=0
msgID=1889535170
message="The startpoint has been set for Sync Pipe 'Sync Pipe1'.
Synchronization will resume from the last change number in the Sync
Source"
Set StartPoint task 2011072109564107 has been successfully completed
```

Restart synchronization at a specific change log event

About this task

Perform the following steps to restart synchronization at a specific event:

Steps

1. Search for a specific change log event from which to restart the synchronization state. On one of the endpoint servers, run `ldapsearch` to search the change log.

```
$ bin/ldapsearch -p 1389
--bindDN "uid=admin,dc=example,dc=com" \
--bindPassword secret \
--baseDN cn=changelog \
--dontWrap
```

```
"(objectclass=*)"
dn: cn=changelog
objectClass: top
objectClass: untypedObject
cn: changelog
dn: changeNumber=1,cn=changelog
objectClass: changeLogEntry
objectClass: top
targetDN: uid=user.13,ou=People,dc=example,dc=com
changeType: modify
changes::
cmVwbGFjZTogcm9vbU51bWJlcgpyb29tTnVtYmVyOiAwMTM4Ci0KcmVwbGFjZTogbW9kaW
ZpZXJzTmFtZQptb2RpZmllcnNOYW1lOiBjbj1EaXJlY3RvcnkgTWFuYWdlcixjbj1Sb290
IEROcyxjbj1jb25maWcKLQpyZXBsYWNlOiBkcy11cGRhdGUtdGltZQpkcy11cGRhdGUtdG
ltZTo6IEFBQUJKZ25OWlUwPQotCgA=
```

```
changenumber: 1
          ... (more output)
dn: changeNumber=2329,cn=changelog
objectClass: changeLogEntry
objectClass: top
targetDN: uid=user.49,ou=People,dc=example,dc=com
changeType: modify
changes::
cmVwbGFjZTogcm9vbU51bWJlcgpyb29tTnVtYmVyOiAwNDMzCi0KcmVwbGFjZTogbW9kaW
ZpZXJzTmFtZQptb2RpZmllcnNOYW1lOiBjbj1EaXJlY3RvcnkgTWFuYWdlcixjbj1Sb290
IEROcyxjbj1jb25maWcKLQpyZXBsYWNlOiBkcy11cGRhdGUtdGltZQpkcy11cGRhdGUtdG
ltZTo6IEFBQUJKZ25OMC84PQotCgA=
changenumber: 2329
```

2. Restart synchronization from change number 2329 using the `realtime-sync` tool. Any event before this change number will not be synchronized to the target endpoint.

```
$ bin/realtime-sync set-startpoint \
  --change-number 2329 \
  --pipe-name "Sync Pipe 1" \
  --bindPassword secret \
  --no-prompt
```

Change the synchronization state by a specific time duration

The following command will start synchronizing data at the state that occurred 2 hours and 30 minutes prior to the current time on External Server 1 for Sync Pipe 1. Any changes made before this time will not be synchronized. Specify days (d), hours (h), minutes (m), seconds (s), or milliseconds (ms).

Use `realtime-sync` with the `--startpoint-rewind` option to set the synchronization state and begin synchronizing at the specified time.

```
$ bin/realtime-sync set-startpoint \
  --startpoint-rewind 2h30m \
  --pipe-name "Sync Pipe 1" \
  --bindPassword secret \
  --no-prompt
```

Schedule a real-time sync as a task

About this task

The `realtime-sync` tool features both an offline mode of operation as well as the ability to schedule an operation to run within PingDataSync Server's process. To schedule an operation, supply LDAP connection options that allow this tool to communicate with the server through its task interface. Tasks can be scheduled to run immediately or at a later time. Once scheduled, tasks can be managed using the `manage-tasks` tool.

Perform the following steps to schedule a synchronization task:

Steps

1. Use the `--start` option with the `realtime-sync` command to schedule a start for the synchronization topology. The following command will set the start time at July 21, 2016 at 12:01:00 AM. The scheduled task can be stopped with the `--stop` subcommand.

```
$ bin/realtime-sync set-startpoint \
  --pipe-name "sun-to-ds-sync-pipe" \
  --port 389 \
  --bindDN "uid=admin,dc=example,dc=com" \
  --bindPassword secret \
```

```
  --start 20150721000100 \
  --no-prompt
```

```
Set StartPoint task 2009072016103807 scheduled to start Jul 21, 2016
12:01:00 AM CDT
```

2. Run the manage-tasks tool to manage or cancel the task.

```
$ bin/manage-tasks --port 7389 \
  --bindDN "uid=admin,dc=example,dc=com" \
  --bindPassword secret
```

Configure the PingDirectory Server backend for synchronizing deletes

About this task

The PingDirectory Server's change log backend's `changelog-deleted-entry-include-attribute` property specifies which attributes should be recorded in the change log entry during a DELETE operation. Normally, PingDataSync Server cannot correlate a deleted entry to the entry on the destination. If a Sync Class is configured with a filter, such as "`include-filter: objectClass=person`," the `objectClass` attribute must be recorded in the change log entry. Special correlation attributes (other than DN), will also need to be recorded on the change log entry to be properly synchronized at the endpoint server.

On each PingDirectory Server backend, use the `dsconfig` command to set the property.

```
$ bin/dsconfig set-backend-prop --backend-name changelog \
  --set changelog-deleted-entry-include-attribute:objectClass
```

If the destination endpoint is an Oracle/Sun DSEE (or Sun DS) server, the Sun DSEE server does not store the value of the user deleting the entry, specified in the modifiers name attribute. It only stores the value of the user who last modified the entry while it still existed.

To set up a Sun DSEE destination endpoint to record the user who deleted the entry, use the Ping Identity Server SDK to create a plugin, as follows:

Steps

1. Update the Sun DSEE schema to include a `deleted-by-syncauxiliary` objectclass. It will only be used as a marker objectclass, and not require or allow additional attributes to be present on an entry.
2. Update the Sun DSEE Retro Change Log plugin to include the `deleted-by-sync auxiliary` objectclass as a value for the `deletedEntryAttrs` attribute.
3. Write an `LDAPSyncDestinationPlugin` script that in the `preDelete()` method modifies the entry that is being deleted to include the `deleted-by-sync` objectclass.
4. Update the Sync Class filter that is excluding changes by the Sync User to also include `(!(objectclass=deleted-by-sync))`.

Configure DN maps

Similar to attribute maps, DN maps define mappings when destination DNs differ from source DNs. These differences must be resolved using DN maps in order for synchronization to successfully take place. For example, the Sync Source could have a DN in the following format:

```
uid=jdoe,ou=People,dc=example,dc=com
```

While the Sync Destination could have the standard X.500 DN format.

DN mappings allow the use of wild cards for DN transformations. A single wild card (*) matches a single RDN component and can be used any number of times. The double wild card (**) matches zero or more RDN components and can be used only once.

> ⓘ **Note:**
>
> If a literal '*' is required in a DN then it must be escaped as `'\2A'`.

The wild card values can be used in the to-dn-pattern attribute using {1} to replace their original index position in the pattern, or {attr} to match an attribute value. For example:

```
*,**,dc=com->{1},ou=012,o=example,c=us
```

For example, using the DN, `uid=johndoe,ou=People,dc=example,dc=com`, and mapping to the target DN, `uid=johndoe,ou=012,o=example,c=us`:

- "`*`" matches one RDN component, `uid=johndoe`
- "`**`" matches zero or more RDN components, `ou=People,dc=example`
- "`dc=com`" matches `dc=com` in the DN.

The DN is mapped to the `{1},ou=012,o=example,c=us`. "`{1}`" substitutes the first wildcard element "`uid=johndoe`", so that the DN is successfully mapped to:

```
uid=johndoe,ou=012,o=example,c=us
```

Regular expressions and attributes from the user entry can also be used in the `to-dn-pattern` attribute. For example, the following expression constructs a value for the `uid` attribute, which is the RDN, out of the initials (first letter of given name and `sn`) and the employee ID (the `eid` attribute) of a user.

```
uid={givenname:/^(.)(.*)/$1/s}{sn:/^(.)(.*)/$1/s}{eid},{2},o=example
```

> ⓘ **Note:**
>
> PingDataSync Server automatically validates any DN mapping prior to applying the configuration.

Configure a DN map by using dsconfig

About this task

A DN map can be configured using `dsconfig`, either with the interactive DN Map menu, or from the command line.

Perform the following to configure a DN map:

Steps

1. Use `dsconfig` to create a DN map for PingDataSync Server.

```
$ bin/dsconfig --no-prompt create-dn-map \
  --map-name nested-to-flattened \
  --set "from-dn-pattern:*,*,dc=example,dc=com" \
  --set "to-dn-pattern:uid={givenname:/^(.)(.*)/\$1/s}{sn:/^(.)(.*)/\$1/s}
(eid},{2},o=example" \
  --port 1389 \
  --bindDN "uid=admin,dc=example,dc=com" \
  --bindPassword secret
```

2. After DN mappings are configured, add the new DN map to a new Sync Class or modify an existing Sync Class.

```
$ bin/dsconfig --no-prompt set-sync-class-prop \
  --pipe-name test-sync-pipe \
  --class-name test-sync-class \
  --set dn-map:test-dn-map \
  --port 389 --bindDN "uid=admin,dc=example,dc=com" \
  --bindPassword secret
```

Configure synchronization with JSON attribute values

PingDataSync Server supports synchronization of attributes that hold JSON objects. The following scenarios are supported:

- Synchronizing a JSON attribute to another JSON attribute - A subset of fields can be synchronized, optionally retaining fields that appear at the destination but not at the source.
- Synchronizing a JSON attribute to a non-JSON attribute - A single field of the JSON value can be extracted with a constructed attribute mapping.
- **Synchronizing a non-JSON attribute to a JSON attribute** - The source value can be escaped so that it ensures the JSON value is properly formatted.
- **Attribute correlation** - A JSON field can be used when correlating a destination entry with a source entry.

The following examples show configuration scenarios based on the LDAP `ubidEmailJSON` attribute, which has fields of `value`, `type`, `primary`, and `verified`:

```
ubidEmailJSON: {"value" : "jsmith@example.com",
                "type" : "home",
                "primary" : true,
                "verified" : true}
```

Synchronize ubidEmailJSON fully

If a source JSON attribute value should be synchronized fully to a destination JSON attribute value, no special configuration is required.

Synchronize a subset of fields from the source attribute

For example, the following configuration can be used to synchronize the `value` and `type` fields of `ubidEmailJSON` from the source to a destination. To synchronize this source value:

```
ubidEmailJSON: {"value" : "jsmith@example.com",
                "type" : "home",
                "primary" : true}
```

to this value at the destination:

```
ubidEmailJSON: {"value" : "jsmith@example.com",
                "type" : "home"}
```

A JSON Attribute configuration object must be created and associated with the Sync Class. This can be done by either explicitly including the fields to synchronize:

```
$ bin/dsconfig create-json-attribute --pipe-name "A to B" \
  --class-name Users \
  --attribute-name ubidEmailJSON \
  --set include-field:type \
  --set include-field:value
```

Or by excluding the fields that should not be synchronized:

```
$ bin/dsconfig create-json-attribute \
  --pipe-name "A to B" \
  --class-name Users \
  --attribute-name ubidEmailJSON \
  --set exclude-field:preferred \
  --set exclude-field:verified
```

If the destination is prepared to only handle a specific subset of fields, then list the fields to include. However, if only a small, known subset of fields from the source should be excluded, then `exclude-field` could be used. In this example, the destination data for the `ubidEmailJSON` attribute will always be a subset of the full data.

> ⓘ **Note:**
>
> A Sync Class can be configured to exclude certain attributes from synchronization. Creating a regular attribute mapping will override this setting, and the attribute will be synchronized. Creating a JSON attribute mapping does not override this setting, and the JSON attribute will not be synchronized. A JSON attribute is not a traditional attribute mapping. It only includes information on the destination attribute name. To work around this, the attribute either needs to be mapped from a source attribute, or have its value constructed.

The following scenario illustrates how the destination can include additional fields that are not present at the source.

Retain destination-only fields

To synchronize changes to the source fields while preserving the value of the `verified` field of the `ubidEmailJSON` attribute at the destination, configure the JSON Attribute as follows:

```
$ bin/dsconfig create-json-attribute \
  --pipe-name "A to B" \
  --class-name Users \
  --attribute-name ubidEmailJSON \
  --set id-field:value \
  --set exclude-field:verified
```

The `verified` field is excluded and `value` is chosen to correlate destination values with source values. For example, given that the source and destination `value` fields match, if the source initially contained:

```
ubidEmailJSON: {"value" : "jsmith@example.com",
                "type" : "home"}
```

and the destination contained:

```
ubidEmailJSON: {"value" : "jsmith@example.com",
                "type" : "home",
                "verified" : true},
```

if the source changed to:

```
ubidEmailJSON: {"value" : "jsmith@example.com",
                "type" : "other"}
```

then the destination would change to:

```
ubidEmailJSON: {"value" : "jsmith@example.com",
                "type" : "other",
                "verified" : true}
```

However, if the source changed to:

```
ubidEmailJSON: {"value" : "john.smith@example.com",
                "type" : "home"}
```

then the destination would be updated to:

```
ubidEmailJSON: {"value" : "john.smith@example.com",
                "type" : "home"}
```

The `verified` field has been dropped because this logically represents a new JSON object rather than an update of an existing one.

Synchronize a field of a JSON attribute into a non-JSON attribute

If the source stores:

```
ubidEmailJSON: {"value" : "jsmith@example.com",
            "type" : "home"}
```

but the destination stores:

```
mail: jsmith@example.com
```

To synchronize changes between these systems, a constructed attribute mapping must be configured:

```
$ bin/dsconfig create-attribute-mapping \
  --map-name "Attribute Map" \
  --mapping-name mail \
  --type constructed \
  --set "value-pattern:{ubidEmailJSON.value}"
```

The `value-pattern` syntax allows attributes to be referenced by placing them in `{}`. JSON fields within the attribute can be referenced by using the syntax `{attribute.field}`. See this property in the Configuration Reference guide, or `dsconfig` tool command help for more information.

After the “Attribute Map” is created, it can be referenced from the Sync Class:

```
$ bin/dsconfig set-sync-class-prop
  --pipe-name "A to B" \
  --class-name Users \
  --set "attribute-map:Attribute Map"
```

ⓘ **Note:**

While LDAP attribute names are not case sensitive, the JSON field names are. By default, errors related to attribute mapping are not logged. To enable error logging, configure the Debug Logger with the following:

```
$ bin/dsconfig set-log-publisher-prop \
  --publisher-name "File-Based Debug Logger" \
  --set enabled:true
```

```
$ bin/dsconfig create-debug-target \
  --publisher-name "File-Based Debug Logger" \
  --target-name com.unboundid.directory.sync.mapping \
  --set debug-level:warning
```

Synchronize a non-JSON attribute into a field of a JSON attribute

This scenario provides a reversal of the previous example. The source stores the following information:

```
mail: jsmith@example.com
```

but the destination stores the following information:

```
ubidEmailJSON: {"value" : "jsmith@example.com"}
```

To construct an attribute value that functions as a JSON object, use *JSON attribute mapping* and *JSON attribute mapping field* configuration objects. The following code provides an example:

```
bin/dsconfig create-attribute-mapping --map-name "Attr Map" \
  --mapping-name ubidEmailJSON \
  --type json

bin/dsconfig create-json-attribute-mapping-field --map-name "Attr Map" \
  --mapping-name ubidEmailJSON \
  --field-name "value" \
  --from-attribute mail
```

For more information about these configuration object types, refer to the configuration reference guide.

You can also use a constructed attribute mapping to construct a JSON attribute value as a raw string, as follows:

```
$ bin/dsconfig create-attribute-mapping \
  --map-name "Attr Map" \
  --mapping-name ubidEmailJSON \
  --type constructed \
  --set 'value-pattern:{{"value" : "{mail:jsonEscape}"}}'
```

When constructing a value, be aware of the following points:

- Double curly brackets (`{{ }}`) are necessary to represent a single curly bracket (`{ }`) in the output. These brackets are typically used to reference attribute values.
- Use the `:jsonEscape` modifier to escape attribute values that appear within a JSON attribute. This step prevents values that include quotes like '`"John Smith" <jsmith@example.com>`' from producing invalid JSON.

In the following example, a JSON Attribute object must be created because the destination value is likely to be augmented with additional information:

```
$ bin/dsconfig create-json-attribute \
  --pipe-name "A to B" \
  --class-name Users \
  --attribute-name ubidEmailJSON \
  --set id-field:value \
  --set include-field:value
```

Synchronize multiple non-JSON attributes into fields of a JSON attribute

Multiple JSON fields can be defined within a single JSON attribute. For any source attribute that does not exist, the corresponding JSON field is omitted from the JSON attribute. The following code demonstrates the mapping of a standard LDAP schema into the standard PingOne user schema.

```
dsconfig create-attribute-map \
  --map-name PingDirectory_to_PingOne_User_Map

...
```

```
dsconfig create-attribute-mapping \
  --map-name PingDirectory_to_PingOne_User_Map \
  -mapping-name name \
  --type json

dsconfig create-json-attribute-mapping-field \
  --map-name PingDirectory_to_PingOne_User_Map \
  --mapping-name name \
  --field-name formatted \
  --set from-attribute:cn \
  --json-type string

dsconfig create-json-attribute-mapping-field \
  --map-name PingDirectory_to_PingOne_User_Map \
  --mapping-name name \
  --field-name given \
  --set from-attribute:givenName \
  --json-type string

dsconfig create-json-attribute-mapping-field \
  --map-name PingDirectory_to_PingOne_User_Map \
  --mapping-name name \
  --field-name family \
  --set from-attribute:sn \
  --json-type string
```

Correlating attributes based on JSON fields

When the destination of a Sync Pipe is a Ping Directory Server or PingDirectoryProxy Server, source and destination entries can be correlated by referencing a field within a JSON attribute. In the following example, source entries will be matched with destination entries that have the same `value` field within the `ubidEmailJSON` value.

```
$ bin/dsconfig set-sync-class-prop \
  --pipe-name "A to B" \
  --class-name Users \
  --set destination-correlation-attributes:ubidEmailJSON.value
```

This could also be used with the previous example, which does not store `ubidEmailJSON.value` at the source but maps into it before correlating at the destination.

Configure fractional replication

About this task

PingDataSync Server supports fractional replication to any server type. For example, if a replica only performs user authentications, PingDataSync Server can be configured to propagate only the `uid` and `userpassword` password policy attributes, reducing the database size at the replica and the network traffic needed to keep this servers synchronized.

The following example configures a fractional replication, where the `uid` and `userPassword` attributes of all entries in the source topology are synchronized to the destination topology. Because the `uid` and `userPassword` attributes are present, the `objectclass` attribute must also be synchronized. The example assumes that PingDataSync Server and the external servers are configured and a Sync Pipe and Sync Class are defined, but real-time synchronization or bulk resync have not been performed.

Perform the following steps to configure fractional replication from the `dsconfig` interactive menu:

Steps

1. On the main menu, type the number corresponding to Sync Classes.
2. On the Sync Class menu, type the number corresponding to viewing and editing an existing Sync Class. Assume that only one Sync Class has been defined.
3. Verify that the Sync Pipe and Sync Class exist.
4. On the SyncClass Properties menu, type the number specifying the source LDAP filter (`include-filter` property) that defines which source entries are to be included in the Sync Class.
5. On the Include-Filter Property menu, type the number corresponding to adding a filter value. For this example, type (`objectclass=person`). When prompted, enter another filter. Press Enter to continue. On the menu, enter 1 to use the value when specifying it.
6. On the SyncClass Properties menu, type the number corresponding to the `auto-mapped-source-attribute` property. Change the value from "`-all-`" to a specific attribute, so that only the specified attribute is automatically mapped from the source topology to the destination topology.
7. On the Auto-Mapped-Source-Attribute Property menu, type the number corresponding to adding the source attributes that will be automatically mapped to the destination attributes of the same name. When prompted, enter each attribute, and then press Enter.

```
Enter another value for the 'auto-mapped-source-attribute' property
[continue]: uid
Enter another value for the 'auto-mapped-source-attribute' property
[continue]: userPassword
Enter another value for the 'auto-mapped-source-attribute' property
[continue]: objectclass
Enter another value for the 'auto-mapped-source-attribute' property
[continue]:
```

8. On the Auto-Mapped-Source-Attribute Property menu, type the number corresponding to removing one or more values. In this example, remove the "`-all-`" value, so that only the `objectclass`, `uid`, and `userPassword` attributes are only synchronized.
9. On the Auto-Mapped-Source-Attribute Property menu, press Enter to accept the values.
10. On the Sync Class Properties menu, type the number corresponding to excluding some attributes from the synchronization process. When using the `objectclass=person` filter, the `cn`, `givenName`, and `sn` attributes must be excluded. Enter the option to add one or more attributes, and then add each attribute to exclude on the `excluded-auto-mapped-source-attributes` Property menu. For this example, exclude the `cn`, and `sn` attributes, which are required attributes of the `Person` objectclass. Also exclude the `givenName` attribute,which is an optional attribute of the `inetOrgPerson` objectclass.

```
Enter another value for the 'excluded-auto-mapped-source-attributes'
property
[continue]: givenName
Enter another value for the 'excluded-auto-mapped-source-attributes'
property
[continue]: sn
Enter another value for the 'excluded-auto-mapped-source-attributes'
property
[continue]:
```

11. On the Excluded-Auto-Mapped-Source-Attributes Property menu, press Enter to accept the changes.

> ⓘ **Note:**
>
> If using `entryUUID` as a correlation attribute, some attribute uniqueness errors may occur while using the `resync` tool. Either set the `excluded-auto-mapped-source-attributes`

property value to `entryUUID` on the Sync Class configuration menu, or run `resync` with the `--excludeDestinationAttr` entryUUID argument.

12. On the Sync Class Properties menu, review the configuration and accept the changes.
13. On the server instances in the destination topology, turn off schema checking to avoid a schema error that occurs when the required attributes in the `Person` objectclass are not present. Make sure that the global configuration property for the `server-group` is set to `all-servers`. Use the following command to turn off schema checking on all of the servers in the group.

```
$ bin/dsconfig --no-prompt set-global-configuration-prop \
  --set check-schema:false \
  --applyChangeTo server-group \
  --port 3389 \
  --bindDN "uid=admin,dc=example,dc=com" \
  --bindPassword secret
```

14. Run `bin/resync` to load the filtered data from the source endpoint to the target endpoint.

```
$ bin/resync --pipe-name "test-sync-pipe" \
  --numPasses 3
```

15. Run `bin/realtime-sync` to start synchronization.

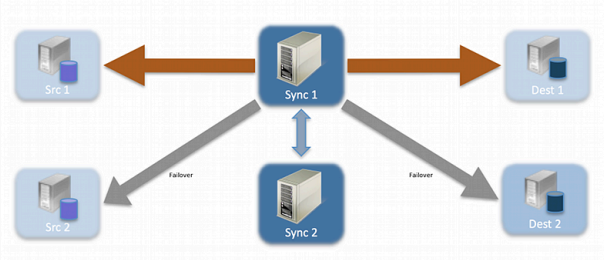
```
$ bin/realtime-sync start --pipe-name "test-sync-pipe" \
  --port 7389 \
  --bindDN "uid=admin,dc=example,dc=com" \
  --bindPassword secret \
  --no-prompt
```

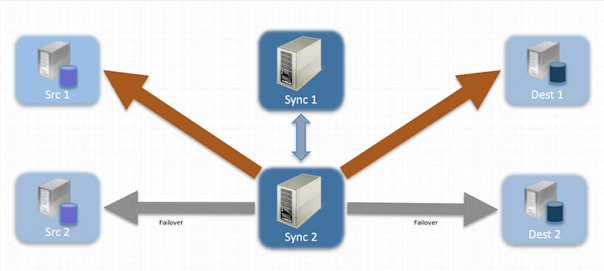
Configure failover behavior

The following illustrates a simplified synchronization topology with a single failover server on the source, destination, and PingDataSync Server, respectively. The gray lines represent possible failover connections in the event the server is down. The external servers are prioritized so that src1 has higher priority than src2; dest1 has higher priority than dest2.

The main PingDataSync Server and its redundant failover instance communicate with each other over LDAP and bind using `cn=IntraSync User,cn=Root DNs,cn=config`. The servers run periodic health checks on each other and share information on all changes that have been processed. Whenever the failover server loses connection to the main server, it assumes that the main server is down and begins processing changes from the last known change. Control reverts back to the main server once it is back online.

Unlike the PingDataSync Servers, the external servers and their corresponding failover server(s) do not run periodic health checks. If an external server goes offline, the failover server will receive transactions and remain connected to PingDataSync Server until the Sync Pipe is restarted, regardless of if the main external server comes back online.





Conditions that trigger immediate failover

Immediate failover occurs when PingDataSync Server receives one of the following error codes from an external server:

- BUSY (51)
- UNAVAILABLE (52)
- SERVER CONNECTION CLOSED (81)
- CONNECT ERROR (91)

If PingDataSync Server attempts a write operation to a target server that returns one of these error codes, PingDataSync Server will automatically fail over to the next highest prioritized server instance in the target topology, issue an alert, and then reissue the retry attempt. If the operation is unsuccessful for any reason, the server logs an error.

Failover server preference

PingDataSync Server supports endpoint failover, which is configurable using the `location` property on the external servers. By default, PingDataSync Server prefers to connect to and failover to endpoint servers in the same location as itself. If no location settings are configured, PingDataSync Server will iterate through the configured list of external servers on the Sync Source and Sync Destination when failing over.

PingDataSync Server does not perform periodic health checks of external servers, and does not failover automatically to a preferred external server. Due to the cost of sync failover, PingDataSync Server remains connected to a given server until the server stops responding or until the Sync Pipe is restarted. When a failover occurs, PingDataSync Server returns to the most preferred server, optionally using location settings to identify it, and works its way down the list. The following provides an example configuration of external servers:

```
austin1.server.com:1389
london1.server.com:2389
boston1.server.com:3389
austin2.server.com:4389
boston2.server.com:5389
london2.server.com:6389
```

If the austin1 server were to become unavailable, PingDataSync Server will automatically pick up changes on the next server on the list, london1. If london1 is also down, then the next server, boston1 will be picked up. After PingDataSync Server iterates through the list, it returns to the top of the list. If PingDataSync Server is connected to london2 and it goes down, it will fail over to austin1.

To minimize WAN traffic, configure the `location` property for each external server using the `dsconfig` command on PingDataSync Server. Assume that PingDataSync Server has its own `location` property (set in the Global Configuration) set to "austin."

```
austin1.server.com:1389 location=austin
london1.server.com:2389 location=london
boston1.server.com:3389 location=boston
austin2.server.com:4389 location=austin
```

```
boston2.server.com:5389 location=boston
london2.server.com:6389 location=london
```

With the `location` property set for each server, PingDataSync Server gets its changes from server austin1. If austin1 goes down, PingDataSync Server will pick up changes from austin2. If austin2 goes down, the server will iterate through the rest of the list in the order it is configured.

The `location` property has another sub-property, `preferred-failover-location` that specifies a set of alternate locations if no servers in this location are available. If multiple values are provided, servers are tried in the order in which the locations are listed. The `preferred-failover-location` property provides more control over the failover process and allows the failover process to jump to a specified location. Care must be used so that circular failover reference does not take place. Here is an example configuration:

```
austin1.server.com:1389 location=austin preferred-failover-location=boston
london1.server.com:2389 location=london preferred-failover-location=austin
boston1.server.com:3389 location=boston preferred-failover-location=london
austin2.server.com:4389 location=austin preferred-failover-location=boston
boston2.server.com:5389 location=boston preferred-failover-location=austin
london2.server.com:6389 location=london preferred-failover-location=london
```

PingDataSync Server will respect the `preferred-failover-location` if it cannot find any external servers in the same location as itself, it will look for any external servers in its own `preferred-failover-location`. In this example, when austin1 becomes unavailable, it will fail over to austin2 because they are in the same location. If austin2 is unavailable, it will fail over to boston1, which is in the `preferred-failover-location` of PingDataSync Server. If boston1 is unavailable, PingDataSync Server will fail over to boston2, and finally, it will try the london1 and london2 servers.

Configuration properties that control failover behavior

There are four important advanced properties to fine tune the failover mechanism:

- `max-operation-attempts` (Sync Pipe)
- `response-timeout` (source and destination endpoints)
- `max-failover-error-code-frequency` (source and destination endpoints)
- `max-backtrack-replication-latency` (source endpoints only)

These properties apply to the following LDAP error codes:

LDAP Error Codes

Error Code	Description
ADMIN_LIMIT_EXCEEDED(11)	Indicates that processing on the requested operation could not continue, because an administrative limit was exceeded.
ALIAS_DEREFERENCING_PROBLEM (36)	Indicates that a problem was encountered while attempting to dereference an alias for a search operation.
CANCELED(118)	Indicates that a cancel request was successful, or that the specified operation was canceled.
CLIENT_SIDE_LOCAL_ERROR(82)	Indicates that a local (client-side) error occurred.
CLIENT_SIDE_ENCODING_ERROR(83)	Indicates that an error occurred while encoding a request.

Error Code	Description
CLIENT_SIDE_DECODING_ERROR(84)	Indicates that an error occurred while decoding a request.
CLIENT_SIDE_TIMEOUT(85)	Indicates that a client-side timeout occurred.
CLIENT_SIDE_USER_CANCELLED(88)	Indicates that a user canceled a client-side operation.
CLIENT_SIDE_NO_MEMORY(90)	Indicates that the client could not obtain enough memory to perform the requested operation.
CLIENT_SIDE_CLIENT_LOOP(96)	Indicates that a referral loop is detected.
CLIENT_SIDE_REFERRAL_LIMIT_EXCEEDED (97)	Indicates that the referral hop limit was exceeded.
DECODING_ERROR(84)	Indicates that an error occurred while decoding a response.
ENCODING_ERROR(83)	Indicates that an error occurred while encoding a response.
INTERACTIVE_TRANSACTION_ ABORTED (30221001)	Indicates that an interactive transaction was aborted.
LOCAL_ERROR(82)	Indicates that a local error occurred.
LOOP_DETECT(54)	Indicates that a referral or chaining loop was detected while processing a request.
NO_MEMORY(90)	Indicates that not enough memory could be obtained to perform the requested operation.
OPERATIONS_ERROR(1)	Indicates that an internal error prevented the operation from being processed properly.
OTHER (80)	Indicates that an error occurred that does not fall into any of the other categories.
PROTOCOL_ERROR(2)	Indicates that the client sent a malformed or illegal request to the server.
TIME_LIMIT_EXCEEDED(3)	Indicates that a time limit was exceeded while attempting to process the request.
TIMEOUT(85)	Indicates that a timeout occurred.
UNWILLING_TO_PERFORM(53)	Indicates that the server is unwilling to perform the requested operation.

The max-operation-attempts property

The `max-operation-attempts` property (part of the Sync Pipe configuration) specifies the maximum number of times to retry a synchronization operation that fails for reasons other than the Sync Destination being busy, unavailable, server connection closed, or connect error.

To change the default number of retries, use `dsconfig` in non-interactive mode to change the `max-operation-attempts` value on the Sync Pipe object. The following command changes the number of maximum attempts from five (default) to four.

```
$ bin/dsconfig set-sync-pipe-prop \
```

```
  --pipe-name "Test Sync Pipe" \
  --set max-operation-attempts:4
```

The response-timeout property

The `response-timeout` property specifies how long PingDataSync Server should wait for a response from a search request to a source server before failing with LDAP result code 85 (client-side timeout). When a client-side timeout occurs, the Sync Source will retry the request according to the `max-failover-error-code-frequency` property before failing over to a different source server and performing the retry. The total number of retries will not exceed the `max-operation-attempts` property defined in the Sync Pipe configuration. A value of zero indicates that there should be no client-side timeout. The default value is one minute.

Assuming a bidirectional topology, the property can be set with `dsconfig` on the Sync Source and Sync Destination, respectively.

```
$ bin/dsconfig set-sync-source-prop \
  --source-name src \
  --set "response-timeout:8 s"
```

```
$ bin/dsconfig set-sync-destination-prop \
  --destination-name U4389 \
  --set "responsetimeout:9 s"
```

The max-failover-error-code-frequency property

The `max-failover-error-code-frequency` property (part of the Sync Source configuration) specifies the maximum time period that an error code can re-appear until it fails over to another server instance. This property allows the retry logic to be tuned, so that retries can be performed once on the same server before giving up and trying another server. The value can be set to zero if there is no acceptable error code frequency and failover should happen immediately. It can also be set to a very small value (such as 10 ms) if a high frequency of error codes is tolerable. The default value is three minutes.

To change the `max-failover-error-code-frequency` property, use `dsconfig` in non- interactive mode to change the property on the Sync Source object. The following command changes the frequency from three minutes to two minutes.

```
$ bin/dsconfig set-sync-source-prop \
  --source-name source1 \
  --set "max-failover-error-code-frequency:2 m"
```

The max-backtrack-replication-latency property

The `max-backtrack-replication-latency` property (part of the Sync Source configuration) sets the time period that PingDataSync Server will look for missed changes in the change log due to replication delays. The property should be set to a conservative upper-bound of the maximum replication delay between two servers in the topology. A value of zero implies that there is no limit on the replication latency. The default value is two hours. PingDataSync Server stops looking in the change log once it finds a change that is older than the maximum replication latency than the last change it processed on the failed server.

For example, after failing over to another server, PingDataSync Server must look through the new server's change log to find the equivalent place to begin synchronizing changes.

Normally, PingDataSync Server can successfully backtrack with only a few queries of the directory, but in some situations, it might have to look further back through the change log to make sure that no changes were missed. Because the changes can come from a variety of sources (replication, synchronization, and over LDAP), the replicated changes between directory servers are interleaved in each change log. When

PingDataSync Server fails over between servers, it has to backtrack to figure out where synchronization can safely pick up the latest changes.

Backtracking occurs until the following:

- The server determines that there is no previous change log state available for any source servers, so it must start at the beginning of the change log.
- The server finds the last processed replication change sequence number (CSN) from the last time it was connected to that replica, if at all. This process is similar to the `set- startpoint` functionality on the `realtime-sync` tool.
- The server finds the last processed replication CSN from every replica that has produced a change so far, and it determines that each change entry in the next oldest batch of changes has already been processed.
- The server finds a change that is separated by more than a certain duration (specified by the `max-backtrack-replication-latency` property) from the most recently processed change.

The following command changes the maximum backtracking from two hours to three hours.

```
$ bin/dsconfig set-sync-source-prop \
  --source-name source1 \
  --set "max-backtrack-replication-latency:3 h"
```

Configure traffic through a load balancer

If a PingData server is sitting behind an intermediate HTTP server, such as a load balancer, a reverse proxy, or a cache, it will log incoming requests as originating with the intermediate HTTP server instead of the client that actually sent the request. If the actual client's IP address should be recorded to the trace log, enable `X-Forwarded-*` handling in both the intermediate HTTP server and the PingData server. See the product documentation for the device type. For PingData servers:

- Edit the appropriate Connection Handler object (HTTPS or HTTP) and set `use-forwarded-headers` to `true`.
- When `use-forwarded-headers` is set to `true`, the server will use the client IP address and port information in the `X-Forwarded-*` headers instead of the address and port of the entity that's actually sending the request, the load balancer. This client address information will show up in logs where one would normally expect it to show up, such as in the `from` field of the HTTP REQUEST and HTTP RESPONSE messages.

Configure authentication with a SASL external certificate

About this task

By default, PingDataSync Server authenticates to the PingDirectory Server using LDAP simple authentication (with a bind DN and a password). However, PingDataSync Server can be configured to use SASL EXTERNAL to authenticate to the PingDirectory Server with a client certificate.

> ⓘ **Note:**
>
> This procedure assumes that PingDataSync Server instances are installed and configured to communicate with the backend PingDirectory Server instances using either SSL or StartTLS.

After the servers are configured, perform the following steps to configure SASL EXTERNAL authentication:

Steps

1. Create a JKS keystore that includes a public and private key pair for a certificate that the PingDataSync Server instance(s) will use to authenticate to the PingDirectory Server instance(s). Run the following command in the instance root of one of the PingDataSync Server instances. When prompted for a

keystore password, enter a strong password to protect the certificate. When prompted for the key password, press ENTER to use the keystore password to protect the private key:

```
$ keytool -genkeypair \
  -keystore config/sync-user-keystore \
  -storetype JKS \
  -keyalg RSA \
  -keysize 2048 \
  -alias sync-user-cert \
  -dname "cn=Sync User,cn=Root DNs,cn=config" \
  -validity 7300
```

2. Create a `config/sync-user-keystore.pin` file that contains a single line that is the keystore password provided in the previous step.
3. If there are other PingDataSync Server instances in the topology, copy the `sync-user-keystore` and `sync-user-keystore.pin` files into the config directory for all instances.
4. Use the following command to export the public component of the user certificate to a text file:

```
$ keytool -export \
  -keystore config/sync-user-keystore \
  -alias sync-user-cert \
  -file config/sync-user-cert.txt
```

5. Copy the `sync-user-cert.txt` file into the `config` directory of all PingDirectory Server instances. Import that certificate into each server's primary trust store by running the following command from the server root. When prompted for the keystore password, enter the password contained in the `config/truststore.pin` file. When prompted to trust the certificate, enter `yes`.

```
$ keytool -import \
  -keystore config/truststore \
  -alias sync-user-cert \
  -file config/sync-user-cert.txt
```

6. Update the configuration for each PingDataSync Server instance to create a new key manager provider that will obtain its certificate from the `config/sync-user-keystore` file. Run the following `dsconfig` command from the server root:

```
$ dsconfig create-key-manager-provider \
  --provider-name "Sync User Certificate" \
  --type file-based \
  --set enabled:true \
  --set key-store-file:config/sync-user-keystore \
  --set key-store-type:JKS \
  --set key-store-pin-file:config/sync-user-keystore.pin
```

7. Update the configuration for each LDAP external server in each PingDataSync Server instance to use the newly-created key manager provider, and also to use SASL EXTERNAL authentication instead of LDAP simple authentication. Run the following `dsconfig` command:

```
$ dsconfig set-external-server-prop \
  --server-name ds1.example.com:636 \
  --set authentication-method:external \
  --set "key-manager-provider:Sync User Certificate"
```

Next steps

After these changes, PingDataSync Server should re-establish connections to the LDAP external server and authenticate with SASL EXTERNAL. Verify that PingDataSync Server is still able to communicate with all backend servers by running the `bin/status` command. All of the servers listed in the "--- LDAP

External Servers ---" section should have a status of `Available`. Review the PingDirectory Server access log can to make sure that the BIND RESULT log messages used to authenticate the connections from PingDataSync Server include `authType="SASL", saslMechanism="EXTERNAL", resultCode=0`, and `authDN="cn=Sync User,cn=RootDNs,cn=config"`.

Configure an LDAPv3 Sync Source

Synchronization can be performed with an LDAP V3-compliant source, such as IBM SDS (Tivoli Directory Server), Oracle Unified Directory, DSEE, or OpenDJ, by configuring a Generic LDAP Sync Source. PingDataSync Server relies on the source server having a `cn=changelog` implementation. If the server does not have a `cn=changelog` implementation, a Server SDK Change Detector extension can be configured to define the change detection criteria that PingDataSync Server should use.

If multiple Generic LDAP Sync Source instances are defined, the order in which they are added is used as a priority order for failover. If server locations are defined, PingDataSync Server will always fail over to servers that are in the same location. If there are multiple Sync Sources in the same location as PingDataSync Server, then PingDataSync Server will fail over to the first local server in the list and proceed down the list.

During synchronization, when a change is detected by PingDataSync Server, the changed entry is fetched from the source. Initially, the DN of the entry is used to search for the entry. If that search fails then a second search is performed using the `unique-id-attribute` if it is defined. This is typically an operational attribute that is automatically generated by the server, such as `entryUUID`.

Server SDK extensions

Custom server extensions can be created with the Server SDK. Extension bundles are installed from a .zip archive or a file system directory. Use the `manage-extension` tool to install or update any extension that is packaged using the extension bundle format. It opens and loads the extension bundle, confirms the correct extension to install, stops the server if necessary, copies the bundle to the server install root, and then restarts the server.

> ⓘ **Note:**
>
> The `manage-extension` tool must be used with Java extensions packaged using the extension bundle format. For more information, see the "Building and Deploying Java-Based Extensions" section of the Server SDK documentation.

The Server SDK enables creating extensions for all PingData servers. Cross-product extensions include:

- Access loggers
- Alert handlers
- Error loggers
- Key Manager providers
- Monitor providers
- Trust Manager providers
- OAuth token handlers
- Manage extension plugins

Synchronize with PingOne for Customers

PingDataSync Server supports PingOne for Customers as a synchronization destination and source for newly created or modified accounts with native password changes between directory servers, relational databases, or other PingOne for Customers systems.

This chapter presents configuration procedures for synchronization between PingDirectory Server, Nokia 8661 Directory Server, or other LDAP source servers or targets with PingOne for Customers.

Prerequisites

Before attempting to synchronize changes to or from a PingOne for Customers environment, make certain the prerequisites in this section are satisfied.

Worker application

A Worker application is an administrator application that can have the same roles as human administrators. You can use Worker applications to create a userless service app that can perform administrator functions. Role assignments determine the functions that the app can perform.

Required grant type

By default, Worker applications are configured with the required Client Credentials grant type. They can also be configured to support additional grant/response types, similar to the other app types.

The Worker application can also perform administrator functions with the role of its user. To accomplish this task, give the app one or more additional grant types, which are used instead of the role assignments.

Required roles

A role is a collection of permissions that can be assigned to a user. Of the many roles that PingOne for Customers includes by default, only the Identity Data Admin role, which manages identities and identity data, is required for the Worker app that you need to create. Permissions center around managing user identities and include functions like creating users, resetting a user's password, and creating, editing, and deleting populations.

Create a Worker application

Before you create a Worker application, make certain you have the following information ready:

- The app name and description
- Redirect URLs for authentication (required for interactive applications only)

Perform the following steps to create a Worker app:

1. At the top of the **Administrator Console**, click **Connections**.
2. Click **Applications**, and then click **+ Application.**
3. From the list of application types, select **Worker**.
4. Click **Configure** to view the **Create App Profile** page.
5. Specify the following information:
   - Application name – Unique identifier for the app.
   - Optional: Description – Brief characterization of the app.
   - Optional: Icon – Pictorial representation of the app. Use a file up to 1MB in JPG, JPEG, GIF, or PNG format.
6. Click **Save and Close**.

   The app is displayed on the **Applications** page.
7. Make note of the OAuth client ID, which appears directly below the name of the app.

   This value is required when creating a PingOne for Customers sync destination or source.
8. From the drop-down list to the right of the app, select **Edit** (Pencil).
9. Click **Configuration**.
10. In the **Basic Configuration** section, make a note of the client secret.

    This value is required when creating a PingOne for Customers sync destination or source.
11. In the **Advanced Configuration** section, make the following selections:
    - For grant type, select **Client Credentials**.
    - For a token endpoint authentication method, select **Client Secret Post**.

12. Click **Save**.
13. In the upper-left corner, click **To Application List**.
14. Enable the app by toggling the corresponding on/off switch.

    The switch appears green when the app is enabled.
15. At the top of the **Administrator Console**, click **Settings**.
16. In the navigation panel to the left, click **Environment**# **>**# **Properties**.
17. Make note of the environment ID.

    This value is required when creating a PingOne for Customers sync destination or source.

For more information, refer to *PingOne for Customers Administration Guide*.

PingOne user resource model

A user resource is a unique identity within PingOne that interacts with the applications and services in the environment to which the user is assigned. Users are associated with an environment and a population, and the service implements directory functions to create, read, update, delete, and search for user resources. For more information, refer to the *Users data model* or to the *PingOne for Customers API Guide*.

The **username** field is required with the PingOne user resource model. This field is a string that specifies the user name, which must be unique within an environment. Limited to 128 characters in length, the **username** must be a well-formed email address or a string of any Unicode letter, mark (like an accent or umlaut), dot, underscore, or hyphen.

Configuring PingOne to use SSO for the PingData Administrative Console

Allow administrative users to single sign-on (SSO) to the PingData admin console from PingOne.

Before you begin

You need the following to complete this process:

- A configured PingData server. This server will host the the PingOne administration console console that is being configured for SSO.
- A PingOne for Customers account. For more information, see *Getting started with PingOne for Customers*.
- Access to the the PingOne administration console console. For more information, see *Using the beta version of My Ping*.

Steps

1. In the PingOne administration console, add a link to the PingOne solutions home page.
   a. In the the PingOne administration console admin console, click **Add Environment**.

      If you're adding PingDirectory Server or PingDataGovernance Server to an existing environment, click the name of an environment, click the **Plus** icon and click **Add** to add PingDirectory.
   b. To create an environment, on the **Create Environment** page, select from **Customers**, **Workforce**, or **Custom**.
   c. Select **PingDirectory** and **PingOne for Customers**.
   d. Click **Next**.
   e. Select `It's already been deployed`.
   f. In the **Enter Admin URL** field, enter `https://<hostname>:<port>/console/login`, replacing the bracketed variables with the PingData server's hostname and HTTP port.
   g. Click **Next**.
   h. In the **Environment Name** field, enter a name for this environment.
   i. Optional: In the **Description** field, enter a description for the environment.
   j. From the **Region** list, select your data center region.
   k. From the **License** list, select the license for this environment.
   l. Click **Finish**.
2. To configure the matching administrator accounts for PingOne and the PingData server, go to the PingOne dashboard for the environment that will be used with the PingData server and repeat the following steps for each PingOne user for whom you want to enable SSO.
   a. In the PingOne administration console, on your environment line, click the **PingOne** icon.
   b. In PingOne, go to **Identities**.
   c. On the line of the administrative user you want to configure, click the **Expand** icon.
   d. Run the following **`dsconfig`** command against the PingData server, replacing the bracketed fields with the values of the administrative user.

      ```
      dsconfig create-root-dn-user --user-name <Username> \
        --set first-name:<Given Name> \
        --set last-name:<Family Name>
      ```
3. Register the Administrative Console with PingOne.
   a. Go to *Add an application - Web application* and follow the instructions in the **Add an OIDC application** subsection.
   b. Enter the application properties as shown in the following table.

Property	Value
Application Name	PingData Administrative Console
Description	Application for the PingData Administrative Console
Redirect URLs	`https://<hostname>:<port>/console/oidc/cb`
Attribute Mapping	'Username' = 'sub'

> ⓘ **Note:**
>
> Fill in the bracketed values in **Redirect URLs** with your PingData server's hostname and HTTP port.

4. Edit the listed properties for the newly created application so that the properties have the values show in the following table, following the instructions in *Edit an application - OIDC* in the PingOne Administration Guide.

Property	Value
Response Type	Code
Grant Type	Authorization Code
Token Endpoint Authentication Method	Client Secret Basic

5. Record the values for the following application properties to use in later steps:
   - Issuer
   - Client ID
   - Client Secret
6. Create a copy of the `PingDirectory/config/sample-dsconfig-batch-files/enable-pingone-admin-console-sso.dsconfig` file, leaving the source file as-is.
7. Open the copy of the file and replace the bracketed values with the values from step 5.
8. Run the file using the following command.

```
dsconfig --batch-file \
    enable-pingone-admin-console-sso-copy.dsconfig \
    --no-prompt
```

9. Click the link to the PingData server from the PingOne solutions home page.
   A PingOne sign on page displays.
10. Sign on using the administrative user credentials.
    The Administrative console index page displays.

Synchronize changes to a PingOne for Customers environment

This section describes the configuration that is necessary to synchronize changes to a PingOne for Customers environment. To view an example configuration, refer to the file `reference-ping-one-sync-destination-configuration.dsconfig`, which is located in the folder named `resources`.

Create a PingOne for Customers sync destination

Before you create a PingOne for Customers sync destination, make certain you have the following information ready:

- Environment ID (*environment-id*)
- OAuth client ID (*oauth-client-id*)
- OAuth client secret (*oauth-client-secret*)

For information about obtaining these values, see "Create a worker application" in *Worker application*.

The following sample creates a PingOne for Customers sync destination.

```
dsconfig create-sync-destination  \
  --destination-name PingOne  \
  --type ping-one-customer  \
  --set api-url:https://api.pingone.com/v1 \
  --set auth-url:https://auth.pingone.com/[PING_ONE_ENV_ID]/as/token \
  --set environment-id:[PING_ONE_ENV_ID]  \
  --set oauth-client-id:[PING_ONE_OAUTH_CLIENT_ID]  \
  --set oauth-client-secret:[PING_ONE_OAUTH_CLIENT_SECRET]
```

Configure JSON attribute mapping

The JSON Attribute Mapping type has been added, including sub-objects (JSON Attribute Mapping Field) that allow you to map individual fields. If a source attribute does not have a value, the corresponding field is omitted.

> ⓘ **Note:** It is recommended that you use JSON attribute mappings rather than constructed attribute mappings.

The following are examples of using JSON attribute mappings:

```
dsconfig create-attribute-mapping \
    --map-name PingDirectory_to_PingOne_User_Map \
    --mapping-name name \
    --type json
```

```
dsconfig create-json-attribute-mapping-field \
    --map-name PingDirectory_to_PingOne_User_Map \
    --mapping-name name \
    --field-name formatted \
    --set from-attribute:cn \
    --json-type string
```

```
dsconfig create-json-attribute-mapping-field \
    --map-name PingDirectory_to_PingOne_User_Map \
    --mapping-name name \
    --field-name given \
    --set from-attribute:givenName \
    --json-type string
```

```
dsconfig create-json-attribute-mapping-field \
    --map-name PingDirectory_to_PingOne_User_Map \
    --mapping-name name \
    --field-name family \
```

Configure constructed attribute mapping

> ⓘ **Note:** It is recommended that you use JSON attribute mappings rather than constructed attribute mappings (see *Configure JSON attribute mapping* on page 1414).

The PingOne User model contains simple JSON attributes like `"title": "Director"` as well as complex JSON objects like `{"name": {"given": "Jane", "family": "Doe"}}`. To ensure accurate processing when you construct attribute mappings that interact with complex objects, construct valid JSON strings and use the command **`jsonEscape`**, as the following example shows.

```
dsconfig create-attribute-mapping \
   --map-name PingDirectory_to_PingOne_User_Map  \
   --mapping-name name  \
   --type constructed  \
   --set 'value-pattern:{{"given":"{givenname:jsonEscape}","family":"
{sn:jsonEscape}"}}'
```

Some attributes in the User resource are operational and cannot be modified by synchronizing data. For more information, refer to the *PingOne for Customers API Guide*.

Correlating entries

The PingOne User Resource model provides an attribute named `externalId`. To ensure that users correlate to the appropriate entry in PingDirectory, map `entryUUID` to this value and configure `externalId` as a destination-correlation-attribute on the Sync class.

Considerations and limitations

This section describes limitations and other constraints to consider when synchronizing changes to a PingOne for Customers environment.

Populations

All PingOne user resources must exist within a population. The PingOne Customers synchronization destination provides the following methods for managing a user's population:

- If a single population is in use, set the configuration attribute `default-population-id` on the sync destination.
- If multiple populations are in use, use a constructed attribute mapping.

The following syntax provides an example.

```
dsconfig create-attribute-mapping \
   --map-name PingDirectory_to_PingOne_User_Map  \
   --mapping-name population  \
   --type constructed  \
   --set 'value-pattern:{{"id":"[DEFAULT_POPULATION_ID]"}}'
```

To set the population properly, make certain to construct a valid JSON object.

Synchronize changes from a PingOne for Customers environment

This section describes the configuration that is necessary to synchronize changes from a PingOne for Customers environment. To view an example configuration, refer to the file `reference-ping-one-sync-source-configuration.dsconfig`, which is located in the folder named `resources`.

Create a PingOne for Customers sync source

Before you create a PingOne for Customers sync source, make certain you have the following information ready:

- Environment ID (*environment-id*)
- OAuth client ID (*oauth-client-id*)
- OAuth client secret (*oauth-client-secret*)

For information about obtaining these values, see "Create a worker application" in *Worker application*.

The following sample creates a PingOne for Customers sync source.

```
dsconfig create-sync-source \
  --source-name PingOne \
  --type ping-one-customer \
  --set api-url:https://api.pingone.com/v1 \
  --set auth-url:https://auth.pingone.com/[PING_ONE_ENV_ID]/as/token \
  --set environment-id:[PING_ONE_ENV_ID] \
  --set oauth-client-id:[PING_ONE_OAUTH_CLIENT_ID] \
  --set oauth-client-secret:[PING_ONE_OAUTH_CLIENT_SECRET]
```

Configure attribute mapping

The process of synchronizing data utilizes the concepts and structures associated with LDAP entries. Ping Identity recommends that you conceptualize the PingOne User Resource model as an LDAP entry when

configuring attribute mappings. Additionally, you might need to use JSON pathing when selecting a value for complex JSON attributes within the user.

```
dsconfig create-attribute-mapping \
  --map-name PingOne_to_PingDirectory_User_Map \
  --mapping-name givenname \
  --type constructed \
  --set "value-pattern:{name.given}"
```

Correlating entries

To ensure that users correlate to the appropriate entry in PingDirectory, map the `id` attribute from the user resource to `entryUUID` in PingDirectory.

Considerations and limitations

This section describes limitations and other constraints to consider when synchronizing changes from a PingOne for Customers environment.

Bidirectional synchronization

If you plan on configuring bidirectional synchronization between PingOne for Customers and PingDirectory, make certain that you satisfy the following conditions:

- Use separate worker apps for the source and destination.
- To prevent the unnecessary duplication of changes, add the client ID of the destination worker app to the `actor-id-to-ignore` configuration attribute of the source.
- To ensure that no attribute mappings are mismatched, modify the reference `dsconfig` batch files.

Password synchronization

PingDataSync Server does not support the synchronizing of passwords from PingOne.

Population management

If your PingOne for Customers environment features a large number of populations, or if you want to limit synchronized users to a specific set of populations, provide one or more `population-to-synchronize` configuration attributes to the source. The name or ID of the population can be used.

Synchronization delay

PingDataSync Server propagates changes throughout PingOne for Customers nearly in real time. However, a delay might occur between the time a change occurs in PingOne and the time it becomes available for PingDataSync Server to synchronize. To help ensure that no changes are missed, a default delay of 5 seconds has been configured within the sync source. For environments of sufficient size or with high rates of change, use the configuration attribute `realtime-sync-polling-offset` on the sync source to increase the delay.

Synchronize with Active Directory Systems

PingDataSync Server supports full synchronization for newly created or modified accounts with native password changes between directory server, relational databases, and Microsoft Active Directory systems.

This chapter presents configuration procedures for synchronization between PingDirectory Server, Nokia 8661 Directory Server, or other LDAP source servers or targets with Microsoft Active Directory systems.

Overview of configuration tasks

To configure synchronization with Active Directory systems, the following tasks are performed:

Enable SSL connections

If synchronizing passwords between systems, synchronization with Microsoft Active Directory systems requires that SSL be enabled on the Active Directory domain controller, so that PingDataSync Server can securely propagate the `cn=Sync User` account password and other user passwords to the target.

Run the create-sync-pipe-config tool

On the Ping Data Sync Server, use the **`create-sync-pipe-config`** tool to configure the Sync Pipes to communicate with the Active Directory source or target.

Configure outbound password synchronization on an PingDirectory Server Sync Source

After running the **`create-sync-pipe-config`** tool, determine if outbound password synchronization from an PingDirectory Server Sync Source is required. If so, enable the Password Encryption component on all PingDirectory Server sources that receive password modifications. The PingDirectory Server uses the Password Encryption component, analogous to the Password Sync Agent component, to intercept password modifications and add an encrypted attribute, `ds-changelog-encrypted-password`, to the changelog entry. The component enables passwords to be synchronized securely to the Active Directory system, which uses a different password storage scheme. The encrypted attribute appears in the change log and is synchronized to the other servers, but does not appear in the entries.

Configure outbound password synchronization on an Active Directory Sync Source

After running the **`create-sync-pipe-config`** tool, determine if outbound password synchronization from an Active Directory Sync Source is required. If so, install the Password Sync Agent (PSA) after configuring PingDataSync Server.

Run the realtime-sync set-startpoint tool

The **`realtime-sync set-startpoint`** command may take several minutes to run, because it must issue repeated searches of the Active Directory domain controller until it has paged through all the changes and receives a cookie that is up-to-date.

Configuring synchronization with Active Directory

About this task
The following procedure configures a one-way Sync Pipe with the Active Directory topology as the Sync Source and a PingDirectory Server topology as the Sync Destination.

Steps

1. From the server-root directory, start PingDataSync Server.

```
$ <server-root>/bin/start-server
```

2. Run the **`create-sync-pipe-config`** tool to set up the initial synchronization topology.

```
$ bin/create-sync-pipe-config
```

3. On the Initial Synchronization Configuration Tool menu, press Enter to continue the configuration.
4. On the Synchronization Mode menu, press Enter to select Standard mode.
5. On the Synchronization Directory menu, select the option for one-way (1) or bidirectional (2) for the synchronization topology.
6. On the Source Endpoint Type menu, enter the option for Microsoft Active Directory.

7. On the Source Endpoint Name menu, type a name for the source server, or accept the default.
8. On the Server Security menu, select the security connection type for the source server.
9. On the Servers menu, enter the host name and listener port for the Source Server, or press Enter to accept the default (port 636). The server will attempt a connection to the server. After adding the first server, add additional servers for the source endpoints, which will be prioritized below the first server.
10. On the Synchronization User Account DN menu, enter the User Account DN for the source servers. The account will be used exclusively by PingDataSync Server to communicate with the source external servers. Enter a User Account DN and password. The User Account DN password must meet the minimum password requirements for Active Directory domains.
11. Set up the Destination Endpoint servers.

Active Directory sync user account

The Sync User created for Active Directory is added to the `cn=Administrators` branch and is given most of a root user's permissions. If this account cannot be secured and there is a need to configure the permissions required by the Sync User, the following are required to perform synchronization tasks.

As a Sync Source, these permissions are needed:

- List contents
- Read all properties
- Read permissions

Deleted items are a special case. For the Sync Server to see deleted entries, the user account must have sufficient access to `cn=Deleted Objects,<domain name>`. Giving access to that DN requires using the **`dsacls`** tool, such as:

```
# Take ownership may be required to make the needed changes.
dsacls "CN=Deleted Objects,DC=example,DC=com" /takeOwnership
```

```
# Give the Sync User generic read permission to the domain.
dsacls "CN=Deleted Objects,DC=example,DC=com" /G "example\SyncUser":GR
```

```
# List the permission for the domain.
dsacls "CN=Deleted Objects,DC=example,DC=com"
```

To revoke all permissions from the Sync User, run the following **`dsacls`** command:

```
dsacls "CN=Deleted Objects,DC=example,DC=com" /R "example\SyncUser"
```

If Active Directory is used as a destination for synchronization, the Sync User account should not be changed.

Preparing external servers

About this task
Perform the following steps to prepare external servers:

Steps

1. After configuring the source and destination endpoints, PingDataSync Server prompts to "prepare" each external server. The process requires trusting the certificate presented to the server, and then testing the connection. If this step is not performed, the process can be completed after configuring the Sync Pipes using the prepare-endpoint-server tool.
2. Configuring this server for synchronization requires manager access. Enter the DN and password of an account capable of managing the external directory server.

3. Enter the maximum age of change log entries. The value is formatted as `[number][time- unit]`, where the time unit format resembles ("8h" for eight hours, "3d" for three days, "1w" for one week). Setting this value caps how long the PingDataSync Server can be offline. A smaller value limits how many changes are stored and is necessary to limit excessive changelog growth in high-modification environments.
4. To prepare another server in the topology, follow the prompts. The previously entered manager credentials can be reused to access additional servers. Repeat the process for each server configured in the system.

Configuring sync pipes and sync classes

About this task
Perform the following steps to configure Sync Pipes and Sync Classes:

Steps

1. On the Sync Pipe Name menu, type a unique name to identify the Sync Pipe, or accept the default.
2. On the Pre-Configured Sync Class Configuration for Active Directory Sync Source menu, enter `yes` to synchronize user CREATE operations, and enter the object class for the user entries at the destination server, or accept the default (user). To synchronize user MODIFY and DELETE operations from Active Directory, enter `yes`.
3. To synchronize passwords from Active Directory, press Enter to accept the default (yes). If synchronizing passwords from Active Directory, install the Ping Identity Password Sync Agent component on each domain controller.
4. To create a DN map for the user entries in the Sync Pipe, enter the base DN for the user entries at the Microsoft Active Directory Sync Source, then enter the base DN for the user entries at the PingDataSync Server Destination.
5. A list of basic attribute mappings from the Microsoft Active Directory Source to the PingDirectory Server destination is displayed. More complex attribute mappings involving constructed or DN attribute mappings must be configured with the **`dsconfig`** tool. The following is a sample mapping.

```
Below is a list of the basic mappings that have been set up for user
entries synchronized from Microsoft Active Directory -> PingDirectory
Server. You can add to or modify this list with any direct attribute
mappings. To set up more complex mappings (such as constructed or DN
attribute mappings), use the 'dsconfig' tool.
1) cn -> cn
2) sn -> sn
3) givenName -> givenName
4) description -> description
5) sAMAccountName -> uid
6) unicodePwd -> userPassword
```

6. Enter the option to add a new attribute mapping. Enter the source attribute, and then enter the destination attribute. The following example maps the `telephoneNumber` attribute (Active Directory) to the `otherTelephone` attribute (PingDirectory Server).

```
Select an attribute mapping to remove, or choose 'n' to add a new one
[Press ENTER to continue]: n
Enter the name of the source attribute: telephoneNumber
Enter the name of the destination attribute: otherTelephone
```

7. If synchronizing group CREATE, MODIFY, and DELETE operations from Active Directory, enter `yes`.
8. Review the basic user group mappings.

9. On the Sync Pipe Sync Class Definitions menu, enter another name for a new Sync Class if required. Repeat steps 2–7 to define this new Sync Class. If no additional Sync Class definitions are required, press Enter to continue.
10. Review the Sync Pipe Configuration Summary, and accept the default ("write configuration"), which records the commands in a batch file (`sync-pipe-cfg.txt`). The batch file can be used to set up other topologies. The following summary shows two Sync Pipes and its associated Sync Classes.

```
>>>> Configuration Summary
  Sync Pipe: AD to PingDirectory Server
    Source: Microsoft Active Directory
      Type: Microsoft Active Directory
      Access Account: cn=Sync
User,cn=Users,DC=adsync,DC=PingIdentity,DC=com
      Base DN: DC=adsync,DC=PingIdentity,DC=com
      Servers: 10.5.1.149:636
    Destination: PingDirectory Server
      Type: PingDirectory Server
      Access Account: cn=Sync User,cn=Root DNs,cn=config
      Base DN: dc=example,dc=com
      Servers: localhost:389
    Sync Classes:
      Microsoft Active Directory Users Sync Class
      Base DN: DC=adsync,DC=PingIdentity,DC=com
      Filters: (objectClass=user)
      DN Map: **,CN=Users,DC=adsync,DC=PingIdentity,DC=com ->{1},ou=users,
      dc=example,dc=com
      Synchronized Attributes: Custom set of mappings are defined
      Operations: Creates,Deletes,Modifies
  Sync Pipe: PingDirectory Server to AD
    Source: PingDirectory Server
      Type: PingDirectory Server
      Access Account: cn=Sync User,cn=Root DNs,cn=config
      Base DN: dc=example,dc=com
      Servers: localhost:389
    Destination: Microsoft Active Directory
      Type: Microsoft Active Directory
      Access Account: cn=Sync
User,cn=Users,DC=adsync,DC=PingIdentity,DC=com
      Base DN: DC=adsync,DC=PingIdentity,DC=com
      Servers: 10.5.1.149:636
    Sync Classes:
       PingDirectory Server Users Sync Class
      Base DN: dc=example,dc=com
      Filters: (objectClass=inetOrgPerson)
      DN Map: **,ou=users,dc=example,dc=com ->{1},CN=Users,DC=adsync,
      DC=PingIdentity,DC=com
      Synchronized Attributes: Custom set of mappings are defined
      Operations: Creates,Deletes,Modifies
```

11. To apply the configuration to the local Ping Data Sync Server instance, type `yes`. The configuration is recorded at `<server-root>/logs/tools/createsync-pipe-config.log`.

Configuring password encryption

About this task

This procedure is required if synchronizing passwords from a PingDirectory Server to Active Directory, or if synchronizing clear text passwords. These steps are not required if synchronizing from Active Directory to a PingData PingDirectory Server, or if not synchronizing passwords.

Steps

1. On the Ping Directory Server that will receive the password modifications, enable the Change Log Password Encryption component. The component intercepts password modifications, encrypts the password and adds an encrypted attribute, `ds-changelog- encrypted-password`, to the change log entry. The encryption key can be copied from the output if displayed, or accessed from the `<serverroot>/bin/sync-pipe-cfg.txt` file.

```
$ bin/dsconfig set-plugin-prop --plugin-name "Changelog Password
Encryption" \
  --set enabled:true \
  --set changelog-password-encryption-key:<key>
```

2. On PingDataSync Server, set the decryption key used to decrypt the user password value in the change log entries. The key allows the user password to be synchronized to other servers that do not use the same password storage scheme.

```
$ bin/dsconfig set-global-sync-configuration-prop \
  --set changelog-password-decryption-key:ej5u9e39pqo68
```

Next steps
Test the configuration or populate data in the destination servers using bulk resync mode. See Using the resync Tool on the Identity Sync Server. Then, use **`realtime-sync`** to start synchronizing the data. See Using the realtime-sync Tool for more information. If synchronizing passwords, install the Password Sync Agent (PSA) on all of the domain controllers in the topology.

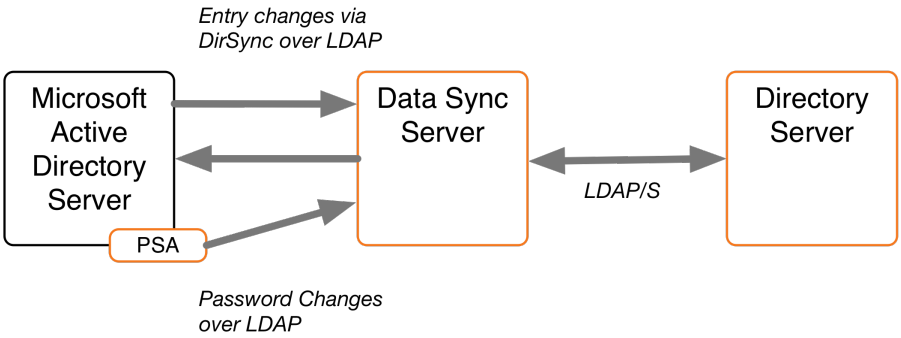
Password sync agent

When synchronizing passwords with Active Directory systems, PingDataSync Server requires that the Ping Identity Password Sync Agent (PSA) be installed on all domain controllers in the synchronization topology. This component provides real-time outbound password synchronization from Microsoft Active Directory to any supported Sync Destinations.

The PSA component provides password synchronization between directories that support differing password storage schemes. The PSA immediately hashes the password with a 160-bit salted secure hash algorithm and erases the memory where the clear-text password was stored. The component only transmits data over a secure (SSL) connection, and follows Microsoft's security guidelines when handling clear-text passwords. The PSA also uses Microsoft Windows password filters, which are part of the local security authority (LSA) process. The password filters enable implementing password policy validation and change notification mechanisms. For more information, see Microsoft's product documentation.

The default password hashing algorithm is SSHA256. To change the algorithm, create a registry key in the Windows registry under `HKLM\SOFTWARE\UnboundID\PasswordSync` called `PASSWORD_HASHING_ALGORITHM`. The options are SSHA, SSHA1, SSHA256, SSHA384, and SSHA512.

> ⓘ **Note:**
>
> For outbound password synchronization from a PingDirectory Server to Active Directory, enable the Password Encryption component. See *Configuring password encryption* on page 1420 for more information.



The PSA supports failover between servers. It caches the hashed password changes in a local database until it can be guaranteed that all PingDataSync Servers in the topology have received them. The failover features enable any or all of the PingDataSync Servers to be taken offline without losing any password changes from Active Directory.

The PSA is safe to leave running on a domain controller indefinitely. To stop synchronizing passwords, remove the `userPassword` attribute mapping in PingDataSync Server, or stop the server. The PSA will not allow its local cache of password changes to grow out of control; it automatically cleans out records from its local database as soon as they have been acknowledged. It also purges changes that have been in the database for more than a week.

Before installing the PSA, consider the following:

- The PSA pre-encodes all passwords with a one-way salted SHA-256 hash before uploading them to PingDataSync. Password changes from Active Directory can only be synchronized to destinations which support setting pre-encoded passwords. Currently, pre-encoded password synchronization is limited to PingDirectory, DSEE, Oracle OUD, and OpenDJ. Active Directory explicitly does not allow for the synchronization of pre-encoded passwords.
- Syncing a pre-encoded password to PingDirectory skips password validation.
- Unlike the changelog password encryption plugin, the PSA never has access to a decryptable version of the password, so it cannot sync it to any source that doesn't support pre-encoded passwords, such as Active Directory.
- The Active Directory Sync Destination drops any passwords it can't decrypt without logging anything about the dropped change.
- Password syncing to Active Directory from either PingDirectory or ActiveDirectory is not possible.
- Make sure that the Active Directory domain controller has SSL enabled and running.
- Make sure the PingDataSync Servers are configured to accept SSL connections when communicating with the Active Directory host.
- At least one Active Directory Sync Source (ADSyncSource) needs to be configured on PingDataSync Server and should point to the domain controller(s) on which the PSA will reside.
- At the time of installation, all PingDataSync Servers in the sync topology must be online and available.
- The PSA component is for outbound-only password synchronization from the Active Directory Systems. It is not necessary if performing a one-way password synchronization from the PingDirectory Server to the Active Directory server.

Install the password sync agent

About this task
The PingDirectory Server distributes the PSA in zip file format with each PingDataSync Server package. The initial installation of the PSA requires a system restart. Perform the following steps to install the PSA:

Steps

1. On the domain controller, double-click the `setup.exe` file to start the installation.
2. Select a folder for the PSA binaries, local database, and log files.
3. Enter the host names (or IP addresses) and SSL ports of the PingDataSync Servers, such as `sync.host.com:636`. Do not add any prefixes to the host names.
4. Enter the Directory Manager DN and password. This creates an ADSync user on PingDataSync Server.
5. Enter a password (secret key) for the ADSync user that will be used by the PSA when connecting to the PingDataSync Server instances.
6. Click **Next** to begin the installation. All of the specified PingDataSync Servers are contacted, and any failures will roll back the installation. If everything succeeds, a message displays indicating that a restart is required. The PSA will start when the computer restarts, and the LSA process is loaded into memory. The LSA process cannot be restarted at runtime.
7. If synchronizing pre-encoded passwords from Active Directory to a Ping Identity system, allow pre-encoded passwords in the default password policy.

```
$ bin/dsconfig set-password-policy-prop \
--policy-name "Default Password Policy" \
--set allow-pre-encoded-passwords:true
```

Upgrade or uninstall the password agent

PingDataSync Server provides the **`update`** tool for upgrades to the server code, including the PSA. The upgrade does not require a restart, because the core password filter is already running under LSA. The upgrade replaces the implementation binaries, which are encapsulated from the password filter DLL.

To uninstall the PSA on the Active Directory system, use **Add/Remove Programs** on the Windows Control Panel. The implementation DLL will be unloaded, and the database and log files are deleted. Only the binaries remain.

The core password filter will still be running under the LSA process. It imposes zero overhead on the domain controller, because the implementation DLL has been unloaded. To remove the password filter itself (located at `C:\WINDOWS\System32\ubidPassFilt.dll`), restart the computer. On restart, the password filter and implementation binaries (in the install folder) can be deleted.

> ⓘ **Note:**
>
> The PSA cannot be reinstalled without another reboot.

Manually configure the password sync agent

Configuration settings for the Password Sync Service are stored in the Windows registry in `HKLM\SOFTWARE\UnboundID\PasswordSync`. Configuration values under this registry key can be modified during runtime. The agent automatically reloads and refreshes its settings from the registry. Verify that the agent is working by checking the current log file, located in `<server- root>\logs\password-sync-current.log`.

Synchronize with Relational Databases

PingDataSync Server supports high-scale, highly-available data synchronization between the directory servers and relational database management systems (RDBMS). Any database with a JDBC 3.0 or later driver can be used.

Use the server SDK

Synchronizing LDAP data to or from a relational database requires creating a JDBC Sync Source or Destination extension to act as an interface between PingDataSync Server and the relational database. The Server SDK provides APIs to develop plugins and third-party extensions to the server using Java or Groovy. The Server SDK's documentation is delivered with the Server SDK build in zip format.

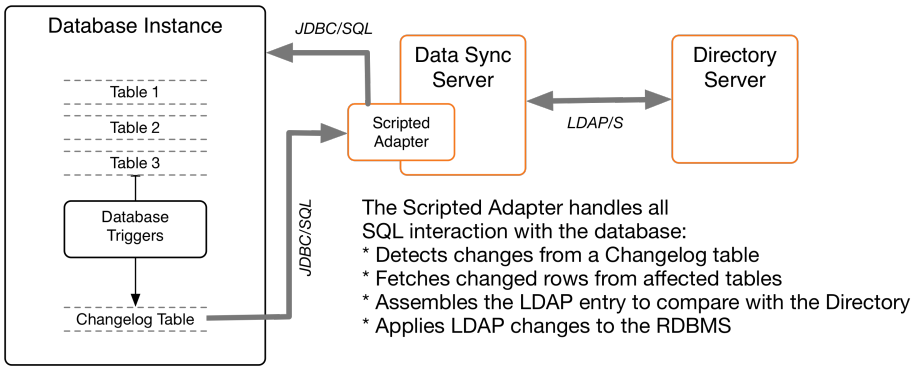
The Server SDK contains two abstract classes that correspond to how the database is used:

```
com.unboundid.directory.sdk.sync.api.JDBCSyncSource

com.unboundid.directory.sdk.sync.api.JDBCSyncDestination
```

The remainder of the SDK contains helper classes and utility functions to facilitate the script implementation. The SDK can use any change tracking mechanism to detect changes in the database. Examples are provided in the `<server-root>/config/jdbc/samples` directory for Oracle Database and Microsoft SQL Server.

PingDataSync Server uses a scripted adapter layer to convert any database change to an equivalent LDAP entry. The Sync Pipe then processes the data through inclusive (or exclusive) filtering using attribute and DN maps defined in the Sync Classes to update the endpoint servers. For example, a script using Java can be configured by setting the `extension-class` property on a `ThirdPartyJDBCSyncSource` or `ThirdPartyJDBCSyncDestination` configuration object within PingDataSync Server. The following is a sample architecture.



RDBMS synchronization process

PingDataSync Server synchronizes data between a directory server and an RDBMS system with a Server SDK extension. PingDataSync Server provides multiple configuration options, such as advanced filtering (fractional and subtree), attribute and DN mappings, transformations, correlations, and configurable logging.

To support synchronizing changes, the database must be configured with a change tracking mechanism. An approach involving triggers, (one trigger per table) to record all changes to a change log table, is recommended. The database change log table Ping Identity should record the type of change (INSERT, UPDATE, DELETE), the specific table name, the unique identifier for the changed row, the database entry type, the changed columns (from the source table), the modifier's name, and the changetimestamp.

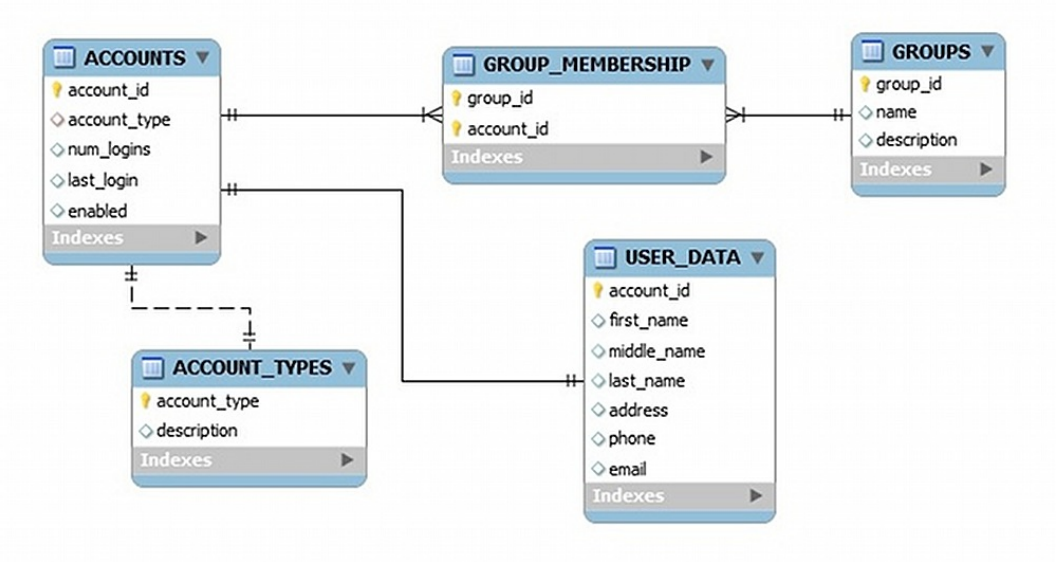
PingDataSync Server delegates the physical interaction with the database to a user- defined extension, which has full control of the SQL queries. The extension layer provides flexibility in how the mapping semantics between the LDAP environment and the relational database environment are defined. The connection management, pooling, retry logic, and other boilerplate code are handled internally by PingDataSync Server.

The RDBMS Synchronization (DBSync) implementation does not support failover between different physical database servers. Most enterprise databases have a built-in failover layer from which PingDataSync Server can point to a single virtual address and port and still be highly available. A single RDBMS node can scale to multiple directory server endpoints.

DBSync example

PingDataSync Server provides a DBSync example between two endpoints consisting of a Ping Identity PingDirectory Server source and a RDBMS system, which will be used in this chapter.

The entity-relational diagram for the normalized database schema is available in `<server-root>/config/jdbc/samples/oracle-db/ComplexSchema.jpg`, and is illustrated here:



Five tables are represented with their primary keys in bold. The entity relations and foreign keys are marked by the relationship lines. The illustration shows mapping to a custom object class on the directory server, while the "groups" table maps to a standard LDAP group entry with `objectclass:groupOfUniqueNames`.

Example directory server entries

The following example assumes that the directory server's schema is configured to handle the mapped attributes. If configuring a database-to-directory Sync Pipe with a newly installed directory server, make sure that the schema has the correct `attributeType` and `objectClass` definitions in place. The definitions can be added in a custom `99-user.ldif` file in the `config/schema` folder of the directory server, if necessary.

The following are the LDAP entries that are used in the synchronization example:

```
dn: accountid=0,ou=People,dc=example,dc=com
objectClass: site-user
firstName: John
lastName: Smith
accountID: 1234
email: jsmith@example.com
phone: +1 556 805 4454
address: 17121 Green Street
numLogins: 4
lastLogin: 20070408182813.196Z
enabled: true

dn: cn=Group 1,ou=Groups,dc=example,dc=com
```

```
objectClass: groupOfUniqueNames
description: This is Group 1
uniqueMember: accountID=0,ou=People,dc=example,dc=com
uniqueMember: accountID=1,ou=People,dc=example,dc=com
```

Configure DBSync

About this task
Configuring a DBSync system includes extra tasks to create the extensions and to configure the database. The overall configuration process is as follows:

Steps

1. Download the appropriate JDBC driver to PingDataSync Server's `/PingDataSync/lib` directory, and restart the server for the driver to load into the runtime.
2. Open the `java.properties` file with a text editor and add the `jdbc.drivers` argument. Save the file. When using custom JDBC endpoints, SQL statements cannot be called unless JDBC drivers are set in the `java.properties` file.
3. Run the **`dsjavaproperties`** command to apply the change. For example, enter the following for start-sync-server:

   ```
   start-sync-server.java-args=-d64 -server -Xmx256m -Xms256m -
   XX:+UseConcMarkSweepGC -
   Djdbc.drivers=foo.bah.Driver:wombat.sql.Driver:com.example.OurDriver ...
   etc.
   ```

4. Create one or more JDBC extensions based on the Server SDK. If configuring for bidirectional synchronization, two scripts are needed: one for the JDBC Sync Source; the other for the JDBC Sync Destination. Place the compiled extensions in the `/lib/extensions` directory.
5. Configure the database change log table and triggers (presented later). The vendor's native change tracking mechanism can be used, but a change log table should also be configured. Each table requires one database trigger to detect the changes and loads them into the change log table.
6. Configure the Sync Pipes, Sync Classes, external servers, DN and attribute maps for one direction.
7. Run the **`resync --dry-run`** command to test the configuration settings.
8. Run **`realtime-sync set-startpoint`** to initialize the starting point for synchronization.
9. Run the **`resync`** command to populate data on the destination endpoint.
10. Start the Sync Pipes using the **`realtime-sync start`** command.
11. Monitor PingDataSync Server using the status commands and logs.
12. For bidirectional synchronization, configure another Sync Pipe, and repeat steps 4–8 to test the system.

Create the JDBC extension

The JDBC extension implementation must be written in Java, or the Groovy scripting language. Consult the Server SDK documentation for details on how to build and deploy extensions. The examples in this guide use Java. Java extensions are more strict and will catch programming errors during compile time rather than at runtime. Groovy is more flexible and can accomplish more with less lines of code.

Groovy scripts must reside in the `/lib/groovy-scripted-extensions` directory (Java implementations reside in `/lib/extensions`), which may also contain other plugins built using the Server SDK. If a script declares a package name, it must live under the corresponding folder hierarchy, just like a Java class. For example, to use a script class called `ComplexJDBCSyncSource` whose package is `com.unboundid.examples.oracle`, place it in `/lib/groovy-scripted-extensions/com/unboundid/examples/oracleand` set the `script- class` property on the Sync Source to `com.unboundid.examples.oracle.ComplexJDBCSyncSource`. There are a few reference implementations provided in the `config/jdbc/samples` directory. Use the **`manage-extension`** tool

in the `bin` directory, or **`bat`** (Windows) to install or update the extension. See the Server SDK extensions section for more information.

When using custom JDBC endpoints, SQL statements cannot be called unless JDBC drivers are set in the `java.properties` file. For example:

```
start-server.java-args=<...> -
Djdbc.drivers=com.microsoft.sqlserver.jdbc.SQLServerDriver
```

The admin must run **`dsjavaproperties -i`** for the update to the `java.properties` file to take effect.

> ⓘ **Note:**
>
> Any changes to an existing script require a manual Sync Pipe restart. Any configuration change automatically restarts the affected Sync Pipe.

The default libraries available on the classpath to the script implementation include:

- Groovy
- LDAP SDK for Java
- JRE

Logging from within a script can be done with the Server SDK's `ServerContext` abstract class. Some of the `ServerContext` methods are not available when the **`resync`** tool runs, because it runs outside of the PingDataSync Server process. Any logging during a **`resync`** operation is saved to the `logs/tools/resync.log` file.

Implement a JDBC sync source

The `JDBCSyncSource` abstract class must be implemented to synchronize data from a relational database. Because PingDataSync Server is LDAP-centric, this class enables database content to be converted into LDAP entries. For more detailed information on the class, see the Server SDK Javadoc.

When using custom JDBC sources, SQL statements cannot be called unless JDBC drivers are set in the `java.properties` file. For example:

```
start-server.java-args=<...> -
Djdbc.drivers=com.microsoft.sqlserver.jdbc.SQLServerDriver
```

The admin must run **`dsjavaproperties -i`** for the update to the `java.properties` file to take effect.

The extension imports classes from the Java API, LDAP SDK for Java API, and the Server SDK. Depending on the data, implement the following methods:

- `initializeJDBCSyncSource` – Called when a Sync Pipe first starts, or when the **`resync`** process starts. Any initialization should be performed here, such as creating internal data structures and setting up variables.
- `finalizeJDBCSyncSource` – Called when a Sync Pipe stops, or when the **`resync`** process stops. Any clean up should be performed here, and all internal resources should be freed.
- `setStartpoint` – Sets the starting point for synchronization by identifying the starting point in the change log. This method should cause all changes previous to the specified start point to be disregarded and only changes after that point to be returned by the `getNextBatchOfChanges` method. There are several different startpoint types (see SetStartpointOptions in the Server SDK), and this implementation is not required to support them all. If the specified startpoint type is unsupported, this method throws an exception (`IllegalArgumentException`). This method can be called from

two different contexts: when the **`realtime-syncset-startpoint`** command is used (the Sync Pipe is required to be stopped) or immediately after a connection is established to the source server.

> ⓘ **Note:**
>
> The `RESUME_AT_SERIALIZABLE` startpoint type must be supported. This method is used when a Sync Pipe first starts and loads its state from disk.

- `getStartpoint` – Gets the current value of the startpoint for change detection.
- `fetchEntry` – Returns a full source entry (in LDAP form) from the database, corresponding to the `DatabaseChangeRecord` object that is passed. The **`resync`** command also uses this class to retrieve entries.
- `acknowledgeCompletedOps`– Provides a means for PingDataSync Server to acknowledge to the database which operations have completed processing.

> ⓘ **Note:**
>
> The internal value for the startpoint should only be updated after a synchronization operation is acknowledged in to this script (through this method). If not, changes could be missed when PingDataSync Server is restarted.

- `getNextBatchOfChanges` – Retrieves the next set of changes for processing. The method also provides a generic means to limit the size of the result set.
- `listAllEntries` – Used by the `resync` command to get a listing of all entries.
- `cleanupChangelog` – In general, we recommend implementing a `cleanupChangelog` method, so that PingDataSync Server can purge old records from the change log table, based on a configurable age.

See the `config/jdbc/samples` directory for example script implementations and the Server SDK Javadoc for more detailed information on each method.

Implement a JDBC sync destination

The `JDBCSyncDestination` abstract class must be implemented to synchronize data into a relational database. The class enables converting LDAP content to database content. The extension imports classes from the Java API, LDAP SDK for Java API, and the Server SDK, depending on the database configuration.

When using custom JDBC destinations, SQL statements cannot be called unless JDBC drivers are set in the `java.properties` file. For example:

```
start-server.java-args=<...> -
Djdbc.drivers=com.microsoft.sqlserver.jdbc.SQLServerDriver
```

The admin must run **`dsjavaproperties -i`** for the update to the `java.properties` file to take effect.

Implement the following methods in the script:

- `initializeJDBCSyncDestination` – Called when a Sync Pipe starts, or when the **`resync`** process starts. Any initialization should be performed here, such as creating internal data structures and setting up variables.
- `finalizeJDBCSyncDestination` – Called when a Sync Pipe stops, or when the **`resync`** process stops. Any clean up should be performed here, and all internal resources should be freed.
- `createEntry` – Creates a full database entry (or row), corresponding to the LDAP entry that is passed in.
- `modifyEntry` – Modifies a database entry, corresponding to the LDAP entry that is passed in.
- `fetchEntry` – Returns a full destination database entry (in LDAP form), corresponding to the source entry that is passed in.

- `deleteEntry` – Deletes a full entry from the database, corresponding to the LDAP entry that is passed in.

For more detailed information on the abstract class, consult the Server SDK Javadoc.

Configure the database for synchronization

About this task
Configuring the database for synchronization includes defining:

- A database SyncUser account
- The change tracking mechanism
- The database triggers (one per table) for the application

The following procedure uses the example setup script in `/config/jdbc/samples/oracle- db/OracleSyncSetup.sql`. Items in brackets are user-named labels.

> ⓘ **Note:**
>
> Database change tracking necessary if synchronizing FROM the database. If synchronizing TO a database, configure the Sync User account and the correct privileges.

Steps

1. Create an Oracle login (`SyncUser`) for PingDataSync Server, so that it can access the database server. Grant sufficient privileges to the `SyncUser` for any tables to be synchronized, and change the default password.

```
CREATE USER SyncUser IDENTIFIED BY password
DEFAULT TABLESPACE users TEMPORARY TABLESPACE temp;
GRANT "RESOURCE" TO SyncUser;
GRANT "CONNECT" TO SyncUser;
```

2. Create change log tables on the database as follows:

```
CREATE TABLE ubid_changelog (
  --This is the unique number for the change change_number Number NOT NULL
PRIMARY KEY,
  --This is the type of change (insert, update, delete). NOTE: This should
represent
  --the actual type of change that needs to happen on the destination(for
example a
  --database delete might translate to a LDAPmodify, etc.)
    change_type VARCHAR2(10) NOT NULL,
  --This is the name of the table that was changed table_name VARCHAR(50)
NOT NULL,
  --This is the unique identifier for the row that was changed. It is up
to
  --the trigger code to construct this, but it should follow a DN-like
format
  --(e.g. accountID={accountID}) where at least the primary key(s) are
  --present. If multiple primary keys are required, they should be
delimited
  --with a unique string, such as '%%' (e.g. accountID={accountID}%%
  --groupID={groupID})
    identifier VARCHAR2(100) NOT NULL,
  --This is the database entry type. The allowable values for this must be
  --set on the JDBC Sync Source configuration within the Synchronization
  --Server.
    entry_type VARCHAR2(50) NOT NULL,
```

```
  --This is a comma-separated list of columns from the source table that
were updated as part of
  --this change.
    changed_columns VARCHAR2(1000) NULL,
  --This is the name of the database user who made the change
    modifiers_name VARCHAR2(50) NOT NULL,
  --This is the timestamp of the change
    change_time TIMESTAMP(3) NOT NULL, CONSTRAINT chk_change_type
    CHECK (change_type IN ('insert','update','delete'))) ORGANIZATION
INDEX;
```

3. Create an Oracle function to get the `SyncUser` name. This is a convenience function for the triggers.

```
CREATE OR REPLACE FUNCTION get_sync_user RETURN VARCHAR2
IS
BEGIN
  RETURN 'SyncUser';
END get_sync_user;
```

4. Create an Oracle sequence object for the change-number column in the change log table.

```
CREATE SEQUENCE ubid_changelog_seq MINVALUE 1 START WITH 1
NOMAXVALUE INCREMENT BY 1 CACHE 100 NOCYCLE;
```

5. Create a database trigger for each table that will participate in synchronization. An example, located in `/config/jdbc/samples/oracle-db/OracleSyncSetup.sql`, shows a trigger for the Accounts table that tracks all changed columns after any INSERT, UPDATE, and DELETE operation. The code generates a list of changed items and then inserts them into the change log table.

Considerations for synchronizing to database destination

When configuring a directory-to-database Sync Pipe, the following are recommended:

Identify the Object Classes

Create a Sync Class per object class, so that they can easily be distinguished and have different mappings and synchronization rules.

For each Sync Class, set the following items in the configuration menus using the `dsconfig` tool.

Set the Include-Filter Property

Make sure the `include-filter` property is set on the Sync Class configuration menu to something that will uniquely identify the source entries, such as `objectClass=customer`.

Create Specific Attribute Mappings

Create an attribute mapping for every LDAP attribute to be synchronized to a database column, add these to a single attribute map, and set it on the Sync Class. With this configured, the script does not need to know about the schema on the directory side. A constructed attribute mapping that maps a literal value to the `objectClass` attribute can be added to the script to determine the database entry type. For example, `"account" -> objectClass` can be added, which would result in the constructed destination LDAP entry always containing an `objectClass` of `"account"`. If needed, a mulit-valued `conditional-value-pattern` property can be used to conditionalize the attribute mapping based on the subtype of the entry or on the value of the attribute. See Conditional Value Mapping for additional information.

Create Specific DN Maps (optional)

If necessary, create a DN map that recognizes the DN's of the source entries and maps them to a desired destination DN. In most cases, the script will use the attributes rather than the DN to figure out which database entry needs to be changed.

Set auto-mapped-source-attribute to "-none-"

Remove the default value of "-all-" from the `auto-mapped-source-attribute` on the Sync Class configuration menu, and replace it with "-none-."

Configure Create-Only Attributes

Any attributes that should be included when created, but never modified (such as `objectclass`) should be specified on the Sync Pipe as a `create-only` attribute. If PingDataSync Server ever computes a difference in that attribute between the source and destination, it will not try to modify it at the destination. To avoid bidirectional loop-back, set the `ignore-changes-by-[user|dn]` property on both Sync Sources when configuring for bidirectional synchronization.

Synchronizing DELETE Operations

On PingDirectory Server and Nokia 8661 Directory Server systems, configure the `changelog-deleted-entry-include-attribute` property on the changelog backend menu using the **`dsconfig`** tool. This property allows for the proper synchronization of DELETE operations. For more information, see Configuring the PingDirectory Server Backend for Synchronizing Deletes .

Attribute-Synchronization-Mode for DBSync

For MODIFY operations, PingDataSync Server detects any change on the source change log, fetches the source entry, applies mappings, computes the equivalent destination entry, fetches the actual destination entry, and then runs a diff between the two entries to determine the minimal set of changes to synchronize. By default, changes on the destination entry are made only for those attributes that were detected in the original change log entry. This is configurable using the `attribute-synchronization-mode` property, which sets the type of diff operation that is performed between the source and destination entries.

If the source endpoint is a database server, set the `attribute-synchronization-mode` property to `all-attributes` on the Sync Class configuration menu. The diff operation will consider all source attributes. Any that have changed will be updated on the destination, even if the change was not originally detected in the change log. This mode is useful when a list of changed columns in the database may not be available. If both endpoints are directory servers, use the default configuration of `modified-attributes-only` to avoid possible replication conflicts.

Handling MODDN Operations

The concept of renaming an entry (modifyDN) does not have a direct equivalent for relational databases. The `JDBCSyncDestination` API does not handle changes of this type. Instead, the `modifyEntry()` method is called as if it is a normal change. The extension can verify if the entry was renamed by looking at the `SyncOperation` that is passed in (`syncOperation.isModifyDN()`). If true, the `fetchedDestEntry` parameter will have the old DN. The new DN can be obtained by calling `syncOperation.getDestinationEntryAfterChange()`.

Configure a directory-to-database sync pipe

The following configures a one-way Sync Pipe with a PingDirectory Server as the Sync Source and an RDBMS (Oracle) system as the Sync Destination with the `create-sync-pipe-config` tool. Sync Pipes can be configured later using **`dsconfig`**.

Create the sync pipe

The following procedures configure the Sync Pipe, external servers, and Sync Classes. The examples are based on the Complex JDBC sample in the `config/jdbc/samples/oracle-db` directory. The **`create-sync-pipe-config`** tool can be run with the server offline and the configuration can later be imported.

1. Run the **`create-sync-pipe-config`** tool.

   ```
   $bin/create-sync-pipe-config
   ```
2. At the `Initial Synchronization Configuration Tool` prompt, press Enter to continue.
3. On the **Synchronization Mode** menu, select **Standard Mode** or **Notification Mode**.
4. On the **Synchronization Directory** menu, choose one-way or bidirectional synchronization.

Configure the sync source

1. On the **Source Endpoint Type** menu, enter the number for the sync source corresponding to the type of source external server.
2. Enter a name for the **Source Endpoint**.
3. Enter the base DN for the directory server, which is used as the base for LDAP searches. For example, enter `dc=example,dc=com`, and then press Enter again to return to the menu. If entering more than one base DN, make sure the DNs do not overlap.
4. On the **Server Security** menu, select the type of communication that PingDataSync Server will use with the endpoint servers.
5. Enter the host and port of the source endpoint server. The Sync Source can specify a single server or multiple servers in a replicated topology. The server tests that a connection can be established.
6. Enter the DN of the Sync User account and create a password for this account. The Sync User account enables PingDataSync Server to access the source endpoint server. By default, the Sync User account is stored as `cn=SyncUser,cn=RootDNs,cn=config`.

Configure the destination endpoint server

1. On the **Destination Endpoint Type** menu, select the type of datastore on the endpoint server. This example is configuring an Oracle Database.
2. Enter a name for the **Destination Endpoint**.
3. On the **JDBC Endpoint Connection Parameters** menu, enter the fully-qualified host name or IP address for the Oracle database server.
4. Enter the listener port for the database server, or press Enter to accept the default (1521).
5. Enter a database name such as `dbsync-test`.
6. The server attempts to locate the JDBC driver in the `lib` directory. If the file is found, a success message is displayed.

   ```
   Successfully found and loaded JDBC driver for:
   jdbc:oracle:thin:@//dbsync-w2k8-vm-2:1521/dbsync-test
   ```

   If the server cannot find the JDBC driver, add it later, or quit the **`create-sync-pipe-config`** tool and add the file to the `lib` directory.
7. Add any additional JDBC connection properties for the database server, or press Enter to accept the default (no). Consult the JDBC driver's vendor documentation for supported properties.
8. Enter a name for the database user account with which PingDataSync Server will communicate, or press Enter to accept the default (SyncUser). Enter the password for the account.
9. On the Standard Setup menu, enter the number for the language (Java or Groovy) that was used to write the server extension.
10. Enter the fully qualified name of the Server SDK extension class that implements the `JDBCSyncDestination` API.

    ```
    Enter the fully qualified name of the Java class that will implement
    com.unboundid.directory.sdk.sync.api.JDBCSyncDestination:
    com.unboundid.examples.oracle.ComplexJDBCSyncDestination
    ```
11. Configure any user-defined arguments needed by the server extension. These are defined in the extension itself and the values are specified in the server configuration. If there are user-defined arguments, enter `yes`.

12. To prepare the Source Endpoint server, which tests the connection to the server with the Sync User account, press **Enter**to accept the default (yes). For the Sync User account, it will return "Denied" as the account has not been written yet to the Directory Server at this time.

```
Testing connection to server1.example.com:1389 Done
Testing 'cn=Sync User,cn=Root DNs,cn=config' access Denied
```

13. To configure the Sync User account on the directory server, press Enter to accept the default (yes). Enter the bind DN (`cn=DirectoryManager`) and the bind DN password of the directory server so that you can configure the `cn=Sync User` account. PingDataSync Server creates the Sync User account, tests the base DN, and enables the change log.

```
Created 'cn=Sync User,cn=Root DNs,cn=config'
Verifying base DN 'dc=example,dc=com' Done
Enabling cn=changelog .....
```

14. Enter the maximum age of the change log entries, or press Enter to accept the default.

Configure the sync pipe and sync classes

The following procedures define a Sync Pipe and two Sync Classes. The first Sync Class is used to match the `accounts` objects. The second Sync Class matches the `group` objects.

1. Continuing from the previous session, enter a name for the Sync Pipe.
2. When prompted to define one or more Sync Classes, enter `yes`.

Configure the accounts Sync Class

1. Enter a name for the Sync Class. For example, type `accounts_sync_class`.
2. If restricting entries to specific subtrees, enter one or more base DNs. If not, press Enter to accept the default (no).
3. To set an LDAP search filter, type `yes` and enter the filter `"(accountid=*)"`. Press Enter again to continue. This property sets the LDAP filters and returns all entries that match the search criteria to be included in the Sync Class. In this example, specify that any entry with an `accountID` attribute be included in the Sync Class. If the entry does not contain any of these values, it will not be synchronized to the target server.
4. Choose to synchronize all attributes, specific attributes, or exclude specific attributes from synchronization, or press Enter to accept the default (all).
5. Specify the operations that will be synchronized for the Sync Class, or press Enter to accept the default.

Configure the groups Sync Class

For this example, configure another Sync Class to handle the `groups` object class. The procedures are similar to that of the configuration steps for the `account_sync_class` Sync Class.

1. On the **Sync Class** menu, enter a name for a new sync class, such as `groups_sync_class`.
2. To restrict entries to specific subtrees, enter one or more base DNs.
3. Set an LDAP search filter. Type `yes` to set up a filter and enter the filter `"(objectClass=groupOfUniqueNames)"`. This property sets the LDAP filters and returns all entries that match the `groupOfUniqueNames` attribute to be included in the Sync Class. If the entry does not contain any of these values, it will not be synchronized to the target server.
4. Choose to synchronize all attributes, specific attributes, or exclude specific attributes from synchronization, or press Enter to accept the default (all).
5. Specify the operations that will be synchronized for the Sync Class, or press Enter to accept the default.
6. At the prompt to enter the name of another Sync Class, press Enter to continue.
7. On the **Default Sync Class Operations** menu, press Enter to accept the default. The Default Sync Class determines how all entries that do not match any other Sync Class are handled.
8. Review the configuration, and press Enter to write the configuration to the server.

Use the **dsconfig** tool to make changes to this configuration. See Configuring PingDataSync Server for configuration options and details.

Considerations for synchronizing from a database source

When synchronizing from a database to a directory or RDBMS server, the following are recommended:

Identify Database Entry Types

Identify the database entry types that will be synchronized, and:

- Set the `database-entry-type` property on the JDBC Sync Source (this is required), and make sure the entry types are what the triggers are inserting into the change tracking mechanism.
- Create a Sync Class per entry type, and set different mappings and rules for each one.

For each Sync Class, do the following:

- Make sure the `include-filter` property is set to match the entry type.
- Create a specific attribute mapping for every database column to be synchronized to an LDAP attribute and set it on the Sync Class. If this is done, the script will not have to know about the schema on the directory side.
- Create a DN map that recognizes the DNs generated by the script and maps them to the correct location at the destination.
- Remove the default value of "-all-" from the `auto-mapped-source-attribute` property on the Sync Class, and replace it with the value `objectClass`. The object class for the fetched source entry is determined by the scripted layer. Values from the database should not be automatically mapped to an attribute with the same name, except the `objectclass` attribute, which should map directly for CREATE operations. If this is not done, an error is generated.
- Change the destination-correlation-attributes property to contain the attributes that uniquely represent the database entries on the directory server destination.

Avoid Bidirectional Loopback

Set the `ignore-changes-by-[user|dn]` property on both Sync Sources when configuring for bidirectional synchronization, to make sure that changes are not looped back by PingDataSync Server.

See Use the create-sync-pipe tool to configure synchronization for details about creating the Sync Pipe.

Synchronize a specific list of database elements

About this task
The **resync** command enables synchronizing a specific set of database keys that are read from a JDBC Sync Source file using the `--sourceInputFile` option. The contents of the file are passed line-by-line into the `listAllEntries()` method of the `JDBCSyncSource` extension, which is used for the Sync Pipe. The method processes the input and returns `DatabaseChangeRecord` instances based on the input from the file.

Perform the following steps to synchronize a specific list of database elements using the **resync** tool:

Steps

1. Create a file of JDBC Sync Source elements. There is no set format for the file, but it typically contains a list of primary keys or SQL queries. For example, create a file containing a list of primary keys and save it as `sourceSQL.txt`.

```
user.0
user.1
user.2
```

```
user.3
```

2. Run the **resync** command with the `--sourceInputFile` option to run on individual primary keys in the file.

```
$ bin/resync --pipe-name "dbsync-pipe" \
--sourceInputFile sourceSQL.txt
```

3. If searching for a specific type of database entry, use the `--entryType` option that matches one of the configured entry types in the JDBCSyncSource.

```
$ bin/resync --pipe-name "dbsync-pipe" \
--entryType account \
--sourceInputFile sourceSQL.txt
```

Synchronize with Apache Kafka

Apache Kafka is an open-source streaming platform that communicates state changes in a distributed environment. Changes to a datastore are sent to a *Kafka topic*, which lets customers update other datastores that contain the same information. Although Kafka persists, orders, and transmits changes, it neither detects changes nor applies them to other datastores.

PingDataSync Server supports Kafka as a sync destination with a Ping Identity sync source, such as PingDirectory Server. In this scenario, the Kafka sync destination publishes changes in PingDirectory Server to a Kafka topic, and displays the entry in JSON format both before and after each change. This level of indirection enables other systems and services to react to changes in PingDirectory Server by consuming messages from the topic.

Endpoints that use the Server SDK do not require custom sync destinations.

Restrictions

The following restrictions affect the manner in which a Kafka sync destination is used:

- The sync source must be a Ping Identity sync source.
- The **changelog** backend must be enabled on all instances of PingDirectory Server.
- To ensure the availability of an entry's attributes from before and after a change, the configuration property **changelog-include-key-attribute** must include `'*'`.
- The Sync Pipe must be configured in Notification mode (**synchronization-mode:notification**) because a Kafka sync destination cannot retrieve the current version of a destination entry.

Configure a Kafka sync destination

Use the **dsconfig** command or the Administration Console to synchronize changes to an Apache Kafka environment. You can reuse existing Ping Identity sync sources that were created for other Sync Pipes.

The following objects are required to configure a Kafka sync destination:

- Kafka cluster external server – Defines the procedure for connecting to a Kafka cluster. The Kafka cluster external server can be referenced from multiple Kafka sync destination configuration objects. The only required property is **bootstrap-server**, which identifies some of the Kafka brokers in the environment.

  When **use-ssl** is set to **true**, the following configuration changes are made:

  - A `trust-manager-provider` is configured to validate the Kafka broker's SSL certificate.
  - A `key-manager-provider` is configured to let the Kafka broker authenticate the PingDataSync Kafka producer.

- Kafka sync destination – References the Kafka cluster external server. The Kafka sync destination must specify the name of the topic to use for publishing messages.

  To adjust Kafka messages beyond the mapping, attribute filtering, and other configuration changes that PingDataSync Server makes, reference one or more of the **`KafkaSyncDestinationPlugin`** extension points that are implemented by using the Server SDK.

Run the **`prepare-endpoint-server`** command for the PingDirectory sync source.

SSL configuration

The following table identifies the **`trust-manager-provider`** and **`key-manager-provider`** properties of the Kafka cluster external server configuration object, as well as the Kafka configuration properties to which they map.

Configuration Object Type	Configuration Property	Kafka Configuration Property
File-based Trust Manager Provider	**`trust-store-file`**	**`ssl.truststore.location`**
File-based Trust Manager Provider	**`trust-store-pin`**, **`trust-store-pin-property`**, **`trust-store-pin-environment-variable`**, or **`trust-store-pin-file`**	**`ssl.truststore.password`**
File-based Key Manager Provider	**`key-store-file`**	**`ssl.keystore.location`**
File-based Key Manager Provider	**`key-store-pin`**, **`key-store-pin-property`**, **`key-store-pin-environment-variable`**, or **`key-store-pin-file`**	**`ssl.keystore.password`**
File-based Key Manager Provider	**`private-key-pin`**, **`private-key-pin-property`**, **`private-key-pin-environment-variable`**, or **`private-key-pin-file`**	**`ssl.key.password`**

Message format

The DN of a changed entry represents the default key for published messages. This value can be overridden in either of the following ways:

- With the **`message-key-attribute`** property of the Kafka sync destination configuration object
- With a **`KafkaSyncDestinationPlugin`** extension point

The value of the of the message is a JSON object that includes the following fields:

- **`type`** – Type of change, such as **`ADD`**, **`MODIFY`**, **`DELETE`**, or **`RESYNC`**.
- **`dn`** – DN of the changed entry.
- **`changeID`** – Unique identifier for the change.
- **`modifiedAttributes`** – List of modified attributes. This field is available only when the type is **`MODIFY`**.
- **`current`** – Entry after the change in JSON format. This field is unavailable when the type is **`DELETE`**.

  The JSON format of a **`current`** entry is compatible with the PingDirectory Directory REST API.
- **`previous`** – Entry before the change in JSON format. This field is unavailable if the type is **`RESYNC`** or **`CREATE`**.

  The JSON format of a **`previous`** entry is compatible with the PingDirectory Directory REST API.

Example ADD

The following code provides an example of an addition:

```
{
  "type": "ADD",
  "dn" : "uid=jsmith,ou=people,dc=example,dc=com",
  "changeID": "changeNumber=1",
  "current": {
    "objectClass": [
      "top",
      "person",
      "organizationalPerson",
      "iNetOrgPerson"
    ],
    "sn": [
      "Smith"
    ],
    "cn": [
      "John Smith"
    ],
    "telephoneNumber": [
      "+1 123 123 1234"
    ],
    "uid": [
      "jsmith"
    ]
  }
}
```

Example MODIFY

The following code provides an example of a modification:

```
{
  "type": "MODIFY",
  "dn" : "uid=jsmith,ou=people,dc=example,dc=com",
  "changeID": "changeNumber=2",
  "modifiedAttributes": [
    "telephoneNumber"
  ],
  "previous": {
    "telephoneNumber": [
      "+1 123 123 1234"
    ],
    "objectClass": [
      "top",
      "person",
      "organizationalPerson",
      "iNetOrgPerson"
    ],
    "sn": [
      "Smith"
    ],
    "cn": [
      "John Smith"
    ],
    "uid": [
      "jsmith"
    ]
  },
  "current": {
    "telephoneNumber": [
```

```
      "+1 321 321 3210"
    ],
    "objectClass": [
      "top",
      "person",
      "organizationalPerson",
      "iNetOrgPerson"
    ],
    "sn": [
      "Smith"
    ],
    "cn": [
      "John Smith"
    ],
    "uid": [
      "jsmith"
    ]
  }
}
```

Example DELETE

The following code provides an example of a deletion:

```
{
  "type": "DELETE",
  "dn" : "uid=jsmith,ou=people,dc=example,dc=com",
  "changeID": "changeNumber=3",
  "previous": {
    "telephoneNumber": [
      "+1 321 321 3210"
    ],
    "objectClass": [
      "top",
      "person",
      "organizationalPerson",
      "iNetOrgPerson"
    ],
    "sn": [
      "Smith"
    ],
    "cn": [
      "John Smith"
    ],
    "uid": [
      "jsmith"
    ]
  }
}
```

Message customization

After PingDataSync Server maps the attributes and filters the results, any entries that are changed will utilize the fields of a Kafka message. To restrict the information that is published to a topic, exclude the unwanted attributes from mapping, or include only the attributes that you want to publish.

By using the **`KafkaSyncDestinationPlugin`** extension point within the Server SDK, you can fully change the message key or value. For more information, including examples, refer to the documentation that is included with the Server SDK packaging.

Synchronize through PingDirectoryProxy servers

Because most data centers deploy directory servers in a proxied environment, PingDataSync Server can also synchronize data through a proxy server in both load-balanced and entry-balancing deployments. The following proxy endpoints are supported:

- Ping Identity PingDirectoryProxy Servers
- Nokia 8661 Directory Proxy Servers

This chapter details a Sync-through-Proxy deployment and provides background information on how it works. Before setting up PingDataSync Server, review the PingDirectoryProxy Server Administration Guide for information about the PingDirectoryProxy Server.

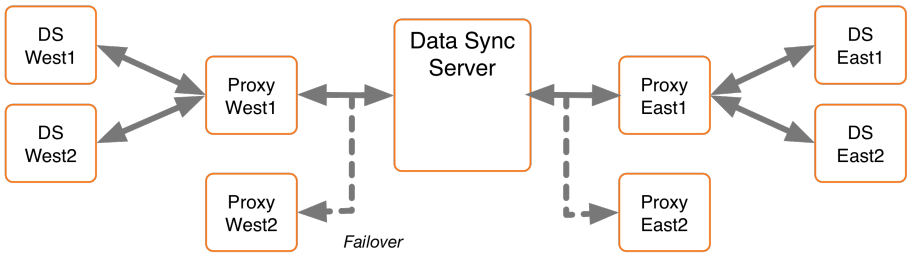
Synchronization through a Proxy Server overview

To handle data synchronization through a proxy server, PingData servers have a `cn=changelog` state management system that supports a token-based API.

In a standard, non-proxied configuration, PingDataSync Server polls the source server for changes, determines if a change is necessary, and fetches the full entry from the source. Then, it finds the corresponding entry in the destination endpoint using correlation rules and applies the minimal set of changes. The server fetches and compares the full entries to make sure it does not synchronize any stale data from the change log.

In a proxied environment, PingDataSync Server passes the request through a proxy server to the backend set of directory servers. PingDataSync Server uses the highest priority proxy server designated in its endpoint server configuration and can use others in the event of a failover.

The following figure illustrates a deployment with two endpoints consisting of a proxy server deployment in front of the backend set of directory servers.



Change log operations

When PingDataSync Server runs a poll for any changes, it sends a get change log batch extended request to the `cn=changelog` backend. The batch request looks for entries in the change log and asks for the server ID, change number, and replica state for each change.

The PingDirectoryProxy Server routes the request to a directory server instance, which then returns a changed entry plus a token identifying the server ID, change number and replica state for each change. The PingDirectoryProxy Server then sends a get change log batch response back to PingDataSync Server with this information. For entry-balancing deployments, the PingDirectoryProxy Server must "re-package" the directory server tokens into its own proxy token to identify the specific data set.

The first time that PingDataSync Server issues the batch request, it also issues a get server ID request to identify the specific server ID that is processing the extended request. The PingDirectoryProxy Server

routes the request to the directory server instance, and then returns a server ID in the response. With the next request, PingDataSync Server sends a 'route to server' request that specifies the server instance to access again in this batch session. It also issues a server ID request in the event that the particular server is down. This method avoids round-robin server selection and provides more efficient overall change processing.

PingDirectory Server and PingDirectoryProxy Server tokens

The PingDirectory Server maintains a change log database index to determine when to resume sending changes (for ADD, MODIFY, or DELETE operations) in its change log. While a simple stand-alone directory server can track its resume point by the last change number sent, replicated servers or servers deployed in entry balancing environments have a different change number ordering in its change log because updates can come from a variety of sources.

The following figure illustrates two change logs in two replicated directory servers, server A and B. "A" represents the replica identifier for a replicated subtree in Server A, and "B" represents the replica identifier for the same replicated subtree in server B. The replica identifiers with a hyphen ("-") mark any local, non-replicated but different changes. While the two replicas record all of the changes, the two change logs have two different change number orderings because updates come in at different times.

Server A			Server B		
ChangeNumber	**ReplicaIdentifier**	**ReplicationCSN**	**ChangeNumber**	**ReplicaIdentifier**	**ReplicationCSN**
1001	A_{ri}	10	2001	B_{ri}	11
1002	-	-	2002	A_{ri}	10
1003	A_{ri}	15	2003	-	-
1004	B_{ri}	11	2004	B_{ri}	12
1005	B_{ri}	12	2005	A_{ri}	15

To track the change log resume position, the PingDirectory Server uses a change log database index to identify the latest change number position corresponding to the highest replication CSN number for a given replica. This information is encapsulated in a directory server token and returned in the get change log batch response to the PingDirectoryProxy Server. The token has the following format:

```
Directory Server Token: server ID, changeNumber,replicaState
```

For example, if the PingDirectoryProxy Server sends a request for any changed entries, and the directory servers return the change number 1003 from server A and change number 2005 from server B, then each directory server token would contain the following information:

```
Directory Server Token A:
  serverID A, changeNumber 1003, replicaState {15(A)}

Directory Server Token B:
  serverID B, changeNumber 2005, replicaState {12(B), 15(A)}
```

Change log tracking in entry balancing deployments

Change log tracking can become more complex in that a shared area of data can exist above the entry-balancing base DN in addition to each backend set having its own set of changes and tokens.

In the following figure, Server A belongs to an entry-balancing set 1, and server B belonging to an entry-balancing set 2. Shared areas that exist above the entry-balancing base DN are assumed to be replicated to all servers. "SA" represents the replica identifier for that shared area on Server A and "SB" represents the replica identifier for the same area on Server B.

Set 1 - Server A			Set 2 - Server B		
ChangeNumber	**ReplicaIdentifier**	**ReplicationCSN**	**ChangeNumber**	**ReplicaIdentifier**	**ReplicationCSN**
1001	SA_{ri}	5	2001	SB_{ri}	10
1002	A_{ri}	10	2002	B_{ri}	20
1003	SB_{ri}	15	2003	SA_{ri}	5

The PingDirectoryProxy Server cannot pass a directory server token from the client to the directory server and back again. In an entry-balancing deployment, the PingDirectoryProxy Server must maintain its own token mechanism that associates a directory server token (`changeNumber`, `replicaIdentifier`, `replicaState`) to a particular backend set.

```
Proxy Token:
backendSetID 1: ds-token 1 (changeNumber, replicaIdentifier, replicaState)
backendSetID 2: ds-token 2 (changeNumber, replicaIdentifier, replicaState)
```

For example, if the PingDirectoryProxy Server returned change 1002 from Server A and change 2002 from Server B, then the Proxy token would contain the following:

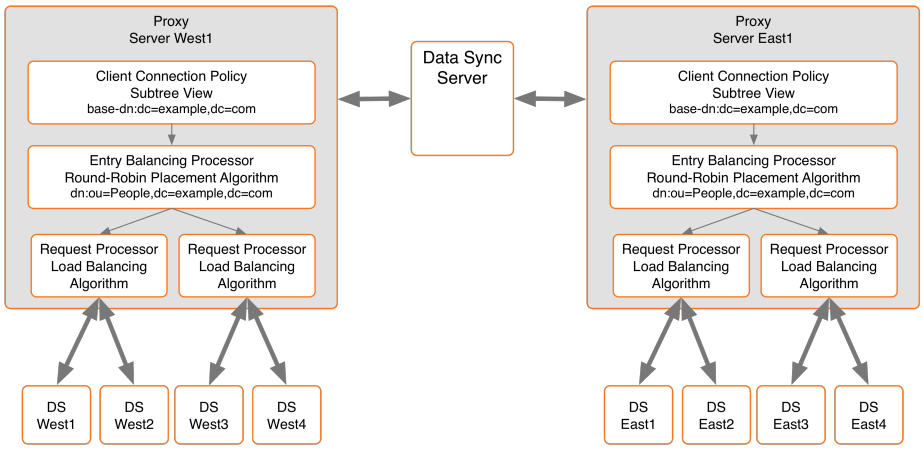
```
Proxy Token:
backendSetID 1: ds-token-1 {serverID A, changeNumber 1002, replicaState (5
(SA), 15(A)}
backendSetID 2: ds-token-2 {serverID B, changeNumber 2002, replicaState (10
(SB), 20(B)}
```

For each change entry returned by a backend, the PingDirectoryProxy Server must also decide whether it is a duplicate of a change made to the backend set above the entry-balancing base. If the change is a duplicate, then it is discarded. Otherwise, any new change is returned with a new value of the proxy token.

Example configuration

This example configures synchronization through a proxy and uses two endpoints consisting of a PingDirectoryProxy Server with a backend set of PingDirectory Servers; both sets are replicated.

The PingDirectoryProxy Server uses an entry-balancing environment for the DN `ou=People,dc=example,dc=com` and provides a subtree view for `dc=example,dc=com` in its client connection policy. For this example, communication is over standard LDAP and failover servers are not installed or designated in PingDataSync Server.



Configure the source PingDirectory Server

About this task
The following procedure configures a backend set of directory servers. The procedure is the same for the source servers and the destination servers in a synchronization topology. For directory server installation and configuration details, see the PingDirectory Server Administration Guide.

Steps

1. On each backend Directory Server that will participate in synchronization, enable the change log database, either from the command line or by using a **`dsconfig`** batch file.

```
$ dsconfig --no-prompt set-backend-prop \
  --backend-name changelog \
  --set enabled:true
```

2. Stop the server if it is running, and import the dataset for the first backend set into the first server in the backend set prior to the import.

```
$ bin/stop-server
$ bin/import-ldif --backendID userRoot --ldifFile ../dataset.ldif
$ bin/start-server
```

3. On the first server instance in the first backend set, configure replication between this server and the second server in the same backend set.

```
$ bin/dsreplication enable --host1 ldap-west-01.example.com \
  --port1 389 \
  --bindDN1 "cn=Directory Manager" \
  --bindPassword1 password \
  --replicationPort1 8989 \
  --host2 ldap-west-02.example.com \
  --port2 389 \
  --bindDN2 "cn=Directory Manager" \
  --bindPassword2 password \
  --replicationPort2 9989 \
  --adminUID admin \
  --adminPassword admin \
  --baseDN dc=example,dc=com \
  --no-prompt
```

4. Initialize the second server in the backend set with data from the first server in the backend set. This command can be run from either instance.

```
$ bin/dsreplication initialize \
  --hostSource ldap-west-01.example.com \
  --portSource 389 \
  --hostDestination ldap-west-02.example.com \
  --portDestination 389 \
  --baseDN "dc=example,dc=com" \
  --adminUID admin \
  --adminPassword admin \
  --no-prompt
```

5. Run the following command to check replica status.

```
$ bin/dsreplication status \
  --hostname ldap-west-01.example.com \
  --port 389 \
  --adminPassword admin \
```

```
  --no-prompt
```

6. Repeat steps 3 through 6 (import, enable replication, initialize replication, check status) for the second backend set.

Configure a proxy server

About this task

The following procedure configures a proxy server, including defining the external servers and configuring the client-connection policy. The procedure is the same for the source servers and the destination servers in a synchronization topology.

For additional changes, use the `dsconfig` tool. For proxy installation and configuration details, see the PingDirectoryProxy Server Administration Guide.

Steps

1. From the PingDirectoryProxy Server root directory, run the **`prepare-external-server`** command to set up the `cn=Proxy User` account for access to the backend directory servers. The server tests the connection and creates the `cn=Proxy User` account.

```
$ bin/prepare-external-server --no-prompt \
  --hostname ldap-west-01.example.com \
  --port 389 --bindDN "cn=Directory Manager" \
  --bindPassword password \
  --proxyBindDN "cn=Proxy User,cn=Root DNs,cn=config" \
  --proxyBindPassword pass \
  --baseDN "dc=example,dc=com"
```

2. Repeat step 1 for any other directory server instances.
3. Run the **`dsconfig`** command to define the external servers and their types. For this example, round-robin load balancing algorithms are defined, which do not require health checks or locations to be specified.

```
$ bin/dsconfig --no-prompt create-external-server \
  --server-name ldap-west-01 \
  --type "ping-identity-ds" \
  --set "server-host-name:ldap-west-01.example.com" \
  --set "server-port:389" \
  --set "bind-dn:cn=Proxy User" \
  --set "password:password" \
  --bindDN "cn=Directory Manager" \
  --bindPassword pxy-pwd
```

```
$ bin/dsconfig --no-prompt create-external-server \
  --server-name ldap-west-02 \
  --type "ping-identity-ds" \
  --set "server-host-name:ldap-west-02.example.com" \
  --set "server-port:389" \
  --set "bind-dn:cn=Proxy User" \
  --set "password:password" \
  --bindDN "cn=Directory Manager" \
  --bindPassword pxy-pwd
```

```
$ bin/dsconfig --no-prompt create-external-server \
  --server-name ldap-west-03 \
  --type "ping-identity-ds" \
  --set "server-host-name:ldap-west-03.example.com" \
  --set "server-port:389" \
```

```
  --set "bind-dn:cn=Proxy User" \
  --set "password:password" \
  --bindDN "cn=Directory Manager" \
  --bindPassword pxy-pwd
```

```
$ bin/dsconfig --no-prompt create-external-server
  --server-name ldap-west-04 \
  --type "ping-identity-ds" \
  --set "server-host-name:ldap-west-04.example.com" \
  --set "server-port:389" \
  --set "bind-dn:cn=Proxy User" \
  --set "password:password" \
  --bindDN "cn=Directory Manager" \
  --bindPassword pxy-pwd
```

4. Create a load-balancing algorithm for each backend set.

```
$ bin/dsconfig --no-prompt create-load-balancing-algorithm \
  --algorithm-name "test-lba-1" \
  --type "round-robin" --set "enabled:true" \
  --set "backend-server:ldap-west-01" \
  --set "backend-server:ldap-west-02" \
  --set "use-location:false" \
  --bindDN "cn=Directory Manager" \
  --bindPassword pxy-pwd
```

```
$ bin/dsconfig --no-prompt create-load-balancing-algorithm \
  --algorithm-name "test-lba-2" \
  --type "round-robin" --set "enabled:true" \
  --set "backend-server:ldap-west-03"
  --set "backend-server:ldap-west-04"
  --set "use-location:false" \
  --bindDN "cn=Directory Manager" \
  --bindPassword pxy-pwd
```

5. Configure the proxying request processors, one for each load-balanced directory server set. A request processor provides the logic to either process the operation directly, forward the request to another server, or hand off the request to another request processor.

```
$ bin/dsconfig --no-prompt create-request-processor \
  --processor-name "proxying-processor-1" --type "proxying" \
  --set "load-balancing-algorithm:test-lba-1" \
  --bindDN "cn=Directory Manager" \
  --bindPassword pxy-pwd
```

```
$ bin/dsconfig --no-prompt create-request-processor \
  --processor-name "proxying-processor-2" --type "proxying" \
  --set "load-balancing-algorithm:test-lba-2" \
  --bindDN "cn=Directory Manager" \
  --bindPassword pxy-pwd
```

6. Define an entry-balancing request processor. This request processor is used to distribute entries under a common parent entry among multiple backend sets. A backend set is a collection of replicated directory servers that contain identical portions of the data. Multiple proxying request processors are used to process operations.

   Next, define the placement algorithm, which selects the server set to use for new add operations to create new entries. In this example, a round-robin placement algorithm forwards LDAP add requests to backend sets.

```
$ bin/dsconfig --no-prompt create-placement-algorithm \
```

```
  --processor-name "entry-balancing-processor" \
  --algorithm-name "round-robin-placement" \
  --set "enabled:true" \
  --type "round-robin" \
  --bindDN "cn=Directory Manager" \
  --bindPassword pxy-pwd
```

7. Define the subtree view that specifies the base DN for the entire deployment.

```
$ bin/dsconfig --no-prompt create-subtree-view \
  --view-name "test-view" \
  --set "base-dn:dc=example,dc=com" \
  --set "request-processor: entry-balancing-processor" \
  --bindDN "cn=Directory Manager" \
  --bindPassword pxy-pwd
```

8. Finally, define a client connection policy that specifies how the client connects to the proxy server.

```
$ bin/dsconfig --no-prompt set-client-connection-policy-prop \
  --policy-name "default" \
  --add "subtree-view:test-view" \
  --bindDN "cn=Directory Manager" \
  --bindPassword pxy-pwd
```

Configure PingDataSync Server

About this task

Configure PingDataSync Server after the PingDirectoryProxy Server and its backend set of PingDirectory Server instances are configured and fully functional for each endpoint, which are labeled as `ldap-west` and `ldap-east` in this example.

For information about installing and configuring PingDataSync Server, see *Installing PingDataSync Server* on page 1297.

Steps

1. From the PingDataSync Server root directory, run the **`create-sync-pipe-config`** tool.

```
$ bin/create-sync-pipe-config
```

2. At the Initial Synchronization Configuration Tool prompt, press Enter to continue.
3. On the Synchronization Mode menu, press Enter to select Standard mode.
4. On the Synchronization Directory menu, choose the option for one-way or bidirectional synchronization.
5. On the First Endpoint Type menu, enter the number for the type of backend data store for the first endpoint. In this example, type the number corresponding to the PingDirectoryProxy Server.

```
>>>> First Endpoint Type
Enter the type of datastore for the first endpoint:
1) Ping Identity Directory Server
2) Ping Identity Directory Proxy Server
3) Alcatel-Lucent Directory Server
4) Alcatel-Lucent Proxy Server
5) Sun Directory Server
6) Microsoft Active Directory
7) Microsoft SQL Server
8) Oracle Database
9) Custom JDBC

b) back
q) quit
```

```
Enter choice [1]: 2
```

6. Enter a descriptive name for the first endpoint.
7. Enter the base DN where PingDataSync Server can search for the entries on the first endpoint server.
8. Specify the type of security when communicating with the endpoint server.
9. Enter the host name and port of the endpoint server. PingDataSync Server tests the connection. Repeat this step if configuring another server for failover.
10. Enter the Sync User account that will be used to access the endpoint server, or press Enter to accept the default `cn=Sync User,cn=Root DNs,cn=config`. Enter a password for the account.
11. The first endpoint deployment is defined using the PingDirectoryProxy Server (`ldap-west`). Repeat steps 5-10 to define the second proxy deployment (`ldap-east`) on PingDataSync Server.
12. Prepare the endpoint servers in the topology. This step confirms that the Sync User account is present on each server and can communicate between PingDataSync Server and the PingDirectoryProxy Servers. In addition to preparing the PingDirectoryProxy Server, PingDataSync Server prepares the backend set of directory servers as the proxy server passes through the authorization to access these servers.
13. Repeat the previous step to prepare the second endpoint server. If other servers have not been prepared, make sure that they are prior to synchronization.
14. Define the Sync Pipe from proxy 1 to proxy 2. In this example, accept the default "Ping Identity Proxy 1 to Ping Identity Proxy 2."
15. To customize on a per-entry basis how attributes get synchronized, define one or more Sync Classes. Create a Sync Class for the special cases, and use the default Sync Class for all other mappings.
16. For the default Sync Class Operations, specify the operations that will be synchronized.
17. Review the configuration settings, and write the configuration to PingDataSync Server in the `sync-pipe-cfg.txt` file.

Test the configuration

About this task

If the `create-sync-pipe-config` tool was not used to create the synchronization configuration, two properties must be verified on each endpoint: `proxy-server` and `use-changelog-batch-request`. The `proxy-server` property should specify the name of the proxy server. The `use-changelog-batch-request` property should be set to `true` on the Sync Source only. The `use-changelog-batch-request` property is not available on the destination endpoint.

The PingDataSync Server connection parameters (hostname, port, bind DN, and bind password) are required.

Steps

1. The following commands check the properties on a Sync Source.

   On the Sync Source:

   ```
   $ bin/dsconfig --no-prompt \
     get-sync-source-prop \
     --source-name "Ping Identity Proxy 1" \
     --property "proxy-server" \
     --property "use-changelog-batch-request"
   ```

   On the Sync Destination:

   ```
   $ bin/dsconfig --no-prompt \
     get-sync-source-prop \
     --source-name "Ping Identity Proxy 2" \
   ```

```
  --property "proxy-server"
```

2. From the server root directory, run the **dsconfig** command to set a flag indicating that the endpoints are PingDirectoryProxy Servers:

```
$ bin/dsconfig --no-prompt \
  set-sync-source-prop \
  --source-name "Ping Identity Proxy 1" \
  --set proxy-server:ldap-west-01 \
  --set use-changelog-batch-request:true
```

```
$ bin/dsconfig --no-prompt \
  set-sync-source-prop \
  --source-name "Ping Identity Proxy 2" \
  --set proxy-server:ldap-east-01
```

3. Run the **resync --dry-run** command to test the configuration settings for each Sync Pipe and debug any issues.

```
$ bin/resync --pipe-name "Ping Identity Proxy 1 to Ping Identity Proxy 2"
--dry-run
```

4. Run **realtime-sync set-startpoint** to initialize the starting point for synchronization.

```
$ realtime-sync set-startpoint --end-of-changelog \
  --pipe-name "Ping Identity Proxy 1 to Ping Identity Proxy 2" \
  --port 389 \
  --bindDN "cn=Directory Manager" \
  --bindPassword password
```

> ⓘ **Note:**
>
> For synchronization through proxy deployments, the `--change-number` option cannot be used with the **realtime-sync set-startpoint** command, because PingDataSync Server cannot retrieve specific change numbers from the backend directory servers. Use `--change-sequence-number`, `--end-of-changelog`, or other options available for the tool.

5. Run the **resync** command to populate data on the endpoint destination server if necessary.

```
$ bin/resync --pipe-name "Ping Identity Proxy 1 to Ping Identity Proxy 2"
\
  --numPasses 3
```

6. Start the Sync Pipe using the **realtime-sync start** command.

```
$ bin/realtime-sync start \
  --pipe-name "Ping Identity Proxy 1 to Ping Identity Proxy 2"
```

7. Monitor PingDataSync Server using the status commands and logs.

Index the LDAP changelog

About this task

The PingData PingDirectory Server and the Nokia 8661 Directory Server (3.0 or later) both support attribute indexing in the changelog backend to enable get changelog batch requests to filter results that include only changes of specific attributes. For example, in an entry balanced proxy deployment, PingDataSync Server sends a get changelog batch request to the Proxy Server, which will send out individual requests to each backend server.

Each directory server that receives a request must iterate over the whole range of changelog entries, and then match entries based on search criteria for inclusion in the batch. The majority of this processing involves determining whether a changelog entry includes changes to a particular attribute or set of attributes, or not. Changelog indexing can dramatically speed up throughput when targeting specific attributes.

Attribute indexing is configured using the `index-include-attribute` and `index-exclude-attribute` properties on the changelog backend. The properties can accept the specific attribute name or special LDAP values "*" to specify all user attributes or "+" to specify all operational attributes.

Perform the following steps to configure changelog indexing:

Steps

1. On all source directory servers, enable changelog indexing for the attributes that will be synchronized. Use the index-include-attribute and index-exclude-attribute properties. The following example specifies that all user attributes (index-include- attribute:*) be indexed in the changelog, except the description and location attributes (index-exclude-attribute:description and index-exclude-attribute:location).

   ```
   $ bin/dsconfig set-backend-prop --backend-name changelog \
     --set "index-include-attribute:*" \
     --set "index-exclude-attribute:description \
     --set "index-exclude-attribute:location
   ```

   ⓘ **Note:**

   There is little performance and disk consumption penalty when using index-include-attribute:* with a combination of `index-exclude-attribute` properties, instead of explicitly defining each attribute using `index-include-attribute`. The only cautionary note about using `index-include-attribute:*` is to be careful that unnecessary attributes get indexed.

2. On PingDataSync Server, configure the `auto-map-source-attributes` property to specify the mappings for the attributes that need to be synchronized.

Results
PingDataSync Server will write a NOTICE message to the error log when the Sync Pipe first starts, indicating whether the server is using changelog indexing or not.

```
[30/Mar/2016:13:21:36.781 -0500] category=SYNC severity=NOTICE
msgID=1894187256 msg="Sync Pipe 'TestPipe' is not using changelog indexing
 on
the source server"
```

Changelog synchronization considerations

If the Sync Source is configured with `use-changelog-batch-request=true`, PingDataSync Server will use the get changelog batch request to retrieve changes from the LDAP changelog. This extended request can contain an optional set of selection criteria, which specifies changelog entries for a specific set of attributes.

PingDataSync Server takes the union of the source attributes from DN mappings, attribute mappings, and the `auto-mapped-source-attributes` property on the Sync Class to create the selection criteria. However, if it encounters the value "-all-" in the `auto-mapped-source-attributes` property, it cannot make use of selection criteria because the Sync Pipe is interested in all possible source attributes.

When the PingDirectory Server receives a get changelog request that contains selection criteria, it returns changelog entries for one or more of the attributes that meet the criteria.

- For ADD and MODIFY changelog entries, the changes must include at least one attribute from the selection criteria.
- For MODDN changelog entries, one of the RDN attributes must match the selection criteria.
- For DELETE changelog entries, one of the `deletedEntryAttrs` much match the selection criteria.

If `auto-mapped` is not set to `all` source attributes, at least one should be configured to show up in the `deletedEntryAttrs` (with the `changelog-deleted-entry-include-attribute` property on the changelog backend).

Another way to do this is to set `use-reversible-form` to `true` on the changelog backend. This includes all attributes in the `deletedEntryAttrs`.

Synchronize in Notification Mode

PingDataSync Server supports a notification synchronization mode that transmits change notifications on a source endpoint to third-party destination applications.

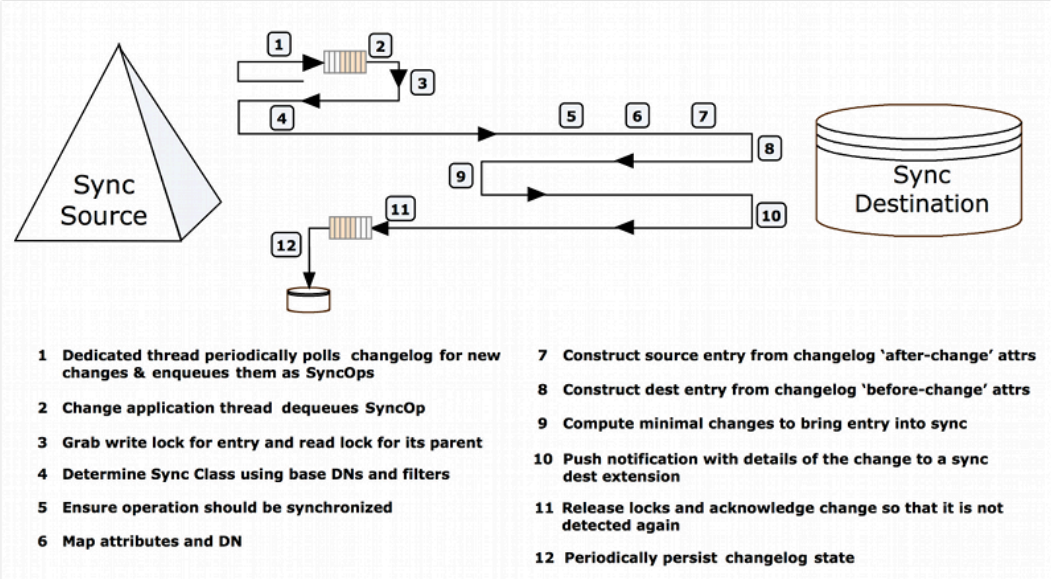
As with standard mode, notifications can be filtered based on the type of entry that was changed, the specific attributes that were changed, and the type of change (ADD, MODIFY, DELETE). PingDataSync Server can send a notification to arbitrary endpoints by using a custom server extension.

Notification mode overview

PingDataSync Server supports standard and notification synchronization modes. Notification Mode polls the directory server's LDAP change log for changes on any entry but skips the fetch and compare phases of processing of Standard Mode. Instead, the Sync Destination is notified of the change regardless of the current state of that entry at the source or destination. PingDataSync Server accesses state information on the change log to reconstruct the before-and-after values of any modified attribute (for example, for MODIFY change operation types). It passes in the change information to a custom server extension based on the Server SDK.

Third-party libraries can be employed to customize the notification message to an output format required by the client application or service. For example, the server extension can use a third-party XML parsing library to convert the change notifications to a SOAP XML format.

Notification mode can only be used with an PingDirectory Server, Nokia 8661 Directory Server, PingDirectoryProxy Server, or Nokia 8661 Directory Proxy Server as the source endpoint.



PingDataSync Server can use notification mode with any type of endpoint; therefore, it is not an absolute requirement to have a custom server extension in your system. For example, it is possible to set up a notification Sync Pipe between two LDAP server endpoints.

Implementation considerations

Before implementing and configuring a Sync Pipe in notification mode, answer the following questions:

- What is the interface to client applications?
- What type of connection logic is required?
- How will the custom server extension handle timeouts and connection failures?
- What are the failover scenarios?
- What data needs to be included in the change log?
- How long do the change log entries need to be available?
- What are the scalability requirements for the system?
- What attributes should be used for correlation?
- What should happen with each type of change?
- What mappings must be implemented?

Use the server SDK and LDAP SDK

To support notification mode, the Server SDK provides a `SyncDestination` extension to synchronize with any client application. The PingDataSync Server engine processes the notification and makes it available to the extension, which can be written in Java or Groovy. This generic extension type can also be used for standard synchronization mode.

Similar to database synchronization, the custom server extension is stored in the `<server-root>/lib/groovy-scripted-extensions` folder (for Groovy-based extensions) or the jar file in the `<server-root>/lib/extensions` folder (for Java-based extensions) prior to configuring PingDataSync Server for notification mode. Groovy scripts are compiled and loaded at runtime.

The Server SDK's `SyncOperation` interface represents a single synchronized change from the Sync Source to the Sync Destination. The same `SyncOperation` object exists from the time a change is detected, through when the change is applied at the destination.

The LDAP SDK's `UnboundIDChangelogEntry` class (in the `com.unboundid.ldap.sdk.unboundidds` package) has high-level methods to work with the `ds-changelog-before-value`, `ds-changelogafter-values`, and `ds-changelog-entry-key-attr-values` attributes. The class is part of the commercial edition of the LDAP SDK for Java and is installed automatically with PingDataSync Server. For detailed information and examples, see the LDAP SDK Javadoc.

Notification mode architecture

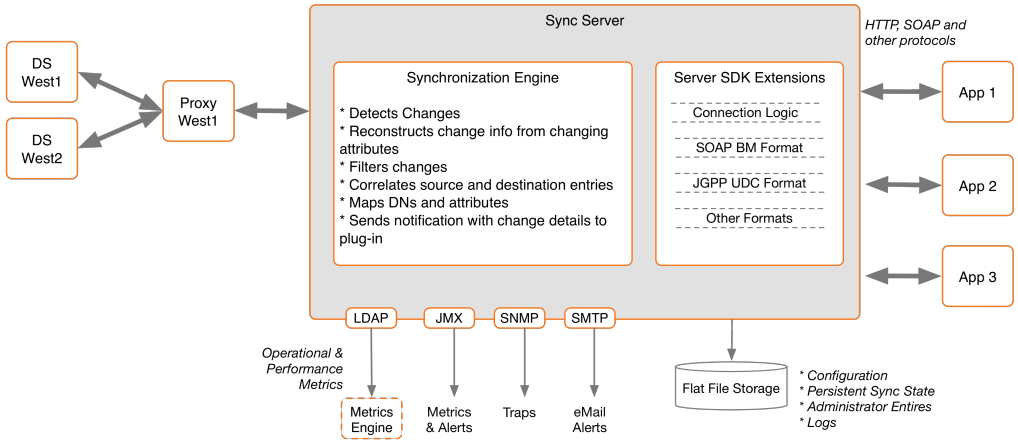
Notification mode, a configuration setting on the Sync Pipe, requires a one-way directional Sync Pipe from a source endpoint topology to a target client application.

PingDataSync Server detects the changes in the PingDirectory Server's LDAP change log, filters the results specified in the Sync Classes, applies any DN and attribute mappings, then reconstructs the change information from the change log attributes. A server extension picks up the notification arguments from the `SyncOperation` interface (part of the Server SDK) and converts the data to the desired output format. The server extension establishes the connections and protocol logic to push the notification information to the client applications or services. All of the operations, administration, and management functions available in standard mode, such as monitoring, (LDAP, JMX, SNMP), alerts (JMX, SNMP, SMTP), and logging features are the same for notification mode.

ⓘ **Note:**

The Server SDK includes documentation and examples on how to create a directory server extension to support notification mode.

For a given entry, PingDataSync Server sends notifications in the order that the changes occurred in the change log even if a modified attribute has been overwritten by a later change. For example, if an entry's `telephoneNumber` attribute is changed three times, three notifications will be sent in the order they appeared in the change log.



Sync source requirements

A separate Sync Pipe is required for each client application that should receive a notification. The Sync sources must consist of one or more instances of the following directory or proxy servers with PingDataSync Server:

- Ping Identity PingDirectory Server and PingDirectoryProxy Server (version 3.0.5 or later)
- Nokia 8661 Directory Server
- The Sync Destination can be of any type

Note:

While the PingDirectoryProxy Server and Nokia 8661 Directory Proxy Server can front other vendor's directory servers, such as Active Directory and Sun DSEE, for processing LDAP operations, PingDataSync Server cannot synchronize changes from these sources through a proxy server. Synchronizing changes directly from Active Directory and Sun DSEE cannot be done with notification mode.

Failover capabilities

For sync source failovers, configure replication between the Directory Servers to ensure data consistency between the servers. A PingDirectoryProxy Server can also front the backend PingDirectory Server set to redirect traffic, if connection to the primary server fails. A PingDirectoryProxy Server must be used for synchronizing changes in an entry-balancing environment. Once the primary PingDirectory Server is online, it assumes control with no information loss as its state information is kept across the backend PingDirectory Servers.

For sync destination failovers, connection retry logic must be implemented in the server extension, which will use the Sync Pipe's advanced property settings to retry failed operations. There is a difference between a connection retry and an operation retry. An extension should not retry operations because PingDataSync Server does this automatically. However, the server extension is responsible for re-establishing connections to a destination that has gone down, or failing over to an alternate server. The server extension can also be designed to trigger its own error- handling code during a failed operation.

For PingDataSync Server failovers, the secondary PingDataSync Servers will be at or slightly behind the state where the primary server initiated a failover. Both primary and secondary PingDataSync Servers track the last failed acknowledgement, so once the primary server fails over to a secondary server, the secondary server will not miss a change.

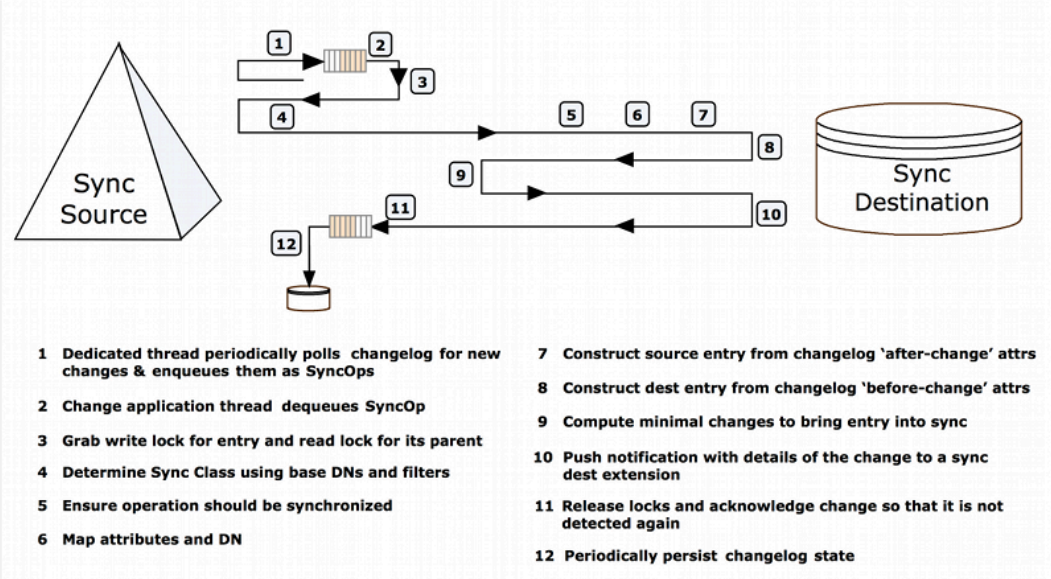
> ⓘ **Note:**
>
> If failover is a concern between PingDataSync Servers, change the `sync-failover-polling-interval` property from 7500 ms to a smaller value. This will result in a quicker failover, but will marginally increase traffic between the two PingDataSync Servers. The `sync-failover-connection-timeout` and `sync-failover-response-timeout` properties may also be updated to use different failover timeout durations. Use **`dsconfig`** to access the property on the Global Sync Configuration menu.

Notification sync pipe change flow

Multi-threaded Sync Pipes allow PingDataSync Server to process multiple notifications in parallel in the same manner as synchronizing changes in standard mode, which increases throughput and offsets network latency. A single change-detection thread pulls in batches of change log entries and queues them internally. To guarantee consistency, PingDataSync Server's internal locking mechanisms ensure the following properties:

- Changes to the same entry will be processed in the same order that they appear in the change log.
- Changes to parent entries will be processed before changes to its children.
- Changes to entries with the same RDN value are handled sequentially.

The number of concurrent threads is configurable on the Sync Pipe using the `num-worker-threads` property. In general, single-threading should be avoided.



Configure notification mode

PingDataSync Server supports notification mode with the following components.

Use the create-sync-pipe-config tool

The **`create-sync-pipe-config`** tool supports the configuration of notification mode. Any pre-existing Sync Sources can be read from the local configuration (in the `config.ldif` file).

LDAP change log features required for notifications

The PingDirectory Server and the Nokia 8661 Directory Server require the following advanced global change log properties: `changelog-max-before-after-values` and `changelog-include-key-attribute`.

These properties are enabled and configured during the `create-sync-pipe-config` configuration process on PingDataSync Server. The properties can also be enabled on the directory servers by using the `dsconfig` advanced properties setting on the Backend Changelog menu.

changelog-include-key-attribute

The `changelog-include-key-attribute` property specifies one or more attributes that should always be included in the change log entry. These are attributes needed to correlate entries between the source and destination, such as `uid`, `employeeNumber`, or `mail`. These properties are also needed for evaluating any filters in the Sync Class. For example, if notifications are only sent for user entries, and the Sync Class included the filter `(objectclass=people)`, the `objectclass` attribute must be configured as a `changelog-include-key-attribute` so that the Sync Pipe can evaluate the inclusion criteria when processing the change. In standard mode, values needed in the filter are read from the entry itself after it is fetched instead of from the changelog entry. These attributes are always included in a change log entry, also called a change record, regardless if they have changed or not.

The `changelog-include-key-attribute` property causes the current (after-change) value of the specified attributes to be recorded in the `ds-changelog-entry-key-attr-values` attribute on the change log entry. This applies for all change types. During a delete operation, the values are from the entry before it was deleted. The key values are recorded on every change and override any settings configured in the `changelog-include-attribute`, `changelog-exclude-attribute`, `changelog-deleted-entry-include-attribute`, or `changelog-deleted-entry- exclude-attribute` properties in the directory server changelog (see the Ping Identity PingDirectoryServer Configuration Reference for more information).

Normal LDAP to LDAP synchronization topologies typically use `dn` as a correlation attribute. If `dn` is used as a correlation attribute only, the `changelog-include-key-attribute` property does not need to be set. However, if another attribute is used for correlation, this property must be set during the Sync Pipe configuration.

The LDAP change log attribute, `ds-changelog-entry-key-attr-values`, stores the attribute that is always included in a change log entry on every change for correlation purposes. In addition to regular attributes, virtual and operational attributes can be specified as entry keys.

To view an example, see the Ping Identity PingDirectory Server Administration Guide.

changelog-max-before-after-values

The `changelog-max-before-after-values` property specifies the maximum number of "before and after" values (default 200) that should be stored for any changed attribute in the change log. Also, when enabled, it will add the `ds-changelog-before-values` and `ds-changelog-after-values` attributes to any change record that contains changes (for Modify and ModifyDN).

The main purpose of the `changelog-max-before-after-values` property is to ensure that an excessively large number of changes is not stored for multi-valued attributes. In most cases, the directory server's schema defines a multi-valued attribute to be unlimited in an entry. For example, if a group entry whose member attribute references 10000 entries, the desire may be to not have all of the attributes if a new member added.

If either the `ds-changelog-before-values` or the `ds-changelog-after-values` attributes exceed the count set in the `changelog-max-before-after-values` property, the attribute values are no longer stored in a change record but its attribute name and number is stored in the `ds-changelog-attr-exceeded-max-values-count` attribute, which appears in the change record.

In addition to this property, set the `use-reversible-form` property to `TRUE`. This guarantees that sufficient information is stored in the change log for all operation types to be able to replay the operations at the destination. The `create-sync-pipe-config` tool configures these properties when it prepares the servers.

The `changelog-max-before-after-values` property configures the following change log attributes:

- `ds-changelog-before-values` – Captures all "before" values of a changed attribute. It will store up to the specified value in the `changelog-max-before-after-values` property (default 200).
- `ds-changelog-after-values` – Captures all "after" values of a changed attribute. It will store up to the specified value in the `changelog-max-before-after-values` property (default 200).
- `ds-changelog-attr-exceeded-max-values-count` – Stores the attribute names and number of before and after values on the change log entry after the maximum number of values (set by the `changelog-max-before-after-values` property) has been exceeded. This is a multi-valued attribute with the following format:

  ```
  attr=attributeName,beforeCount=200,afterCount=201
  ```

  where `attributeName` is the name of the attribute and the `beforeCount` and `afterCount` are the total number of values for that attribute before and after the change, respectively. In either case (before or after the change), if the number of values is exceeding the maximum, those values will not be stored.

LDAP change log for Notification and Standard Mode

Both Notification and Standard mode Sync Pipes can consume the same LDAP Change Log without affecting the other. Standard mode polls the change record in the change log for any modifications, fetches the full entries on the source and the destination, and then compares them for the specific changes. Notification mode gets the before and after values of a changed attribute to reconstruct an entry, and bypasses the fetch-and-compare phase. Both can consume the same LDAP Change Log with no performance loss or conflicts.

> ⓘ **Note:**
>
> If the configuration obtains the change log through a PingDirectoryProxy Server, the contents of the change log will not change as it is being read from the change logs on the directory server backend.

Implementing the server extension

Notification mode relies heavily on the server extension code to process and transmit the change using the required protocol and data formats needed for the client applications.

Create the extension using the Server SDK, which provides the APIs to develop code for any destination endpoint type. The Server SDK's documentation (Javadoc and examples) is delivered with the Server SDK built-in zip format. The SDK provides all of the necessary classes to extend the functionality of PingDataSync Server without code changes to the core product. Once the server extension is in place, use other third-party libraries to transform the notification to any desired output format.

Consider the following when implementing the extension:

Use the manage-extension tool

Use the manage-extension tool in the `bin` directory or `bat` directory (Windows) to install or update the extension. See Managing Extensions for more information.

Review the Server SDK package

Review Server SDK documentation and examples before building and deploy a Java or Groovy extension.

Connection and protocol logic

The Server SDK extension must manage the notification connection and protocol logic to the client applications.

Implementing extensions

Test the create methods, the delete methods, and the modify methods for each entry type. Update the configuration as needed. Finally, package the extensions for deployment. Logging levels can be increased to include more details.

Use the SyncOperation type

The `SyncOperation` class encapsulates everything to do with a given change. Objects of this type are used in all of the synchronization SDK extensions. See the Server SDK Javadoc for the `SyncOperation` class for information on the full set of methods.

Use the EndpointException type

The Sync Destination type offers an exception type called `EndpointException` to extend a standard Java exception and provide custom exceptions. There is also logic to handle LDAP exceptions, using the LDAP SDK.

About the PostStep result codes

The `EndpointException` class uses PostStep result codes that are returned in the server extension:

- `retry_operation_limited` – Retry a failed attempt up to the limit set by `max_operation_attempts`.
- `retry_operation_unlimited` – Retry the operation an unlimited number of times until a success, abort, or `retried_operation_limited`.This should only be used when the destination endpoint is unavailable.
- `abort_operation` – Abort the current operation without any additional processing.

Use the ServerContext class for logging

The `ServerContext` class provides several logging methods which can be used to generate log messages and/or alerts from the scripted layer: `logMessage()`, `sendAlert()`, `debugCaught()`, `debugError()`, `debugInfo()`, `debugThrown()`, `debugVerbose()`, and `debugWarning()`. These are described in the Server SDK API Javadocs. Logging related to an individual `SyncOperation` should be done with the `SyncOperation#logInfo` and `SyncOperation#logError` methods.

Diagnosing script errors

When a Groovy extension does not behave as expected, first look in the error log for stack traces. If `classLoader` errors are present, the script could be in the wrong location or may not have the correct package. Groovy checks for errors at runtime. Business logic errors must be systematically found by testing each operation. Make sure logger levels are set high enough to debug.

Configure the Notification sync pipe

The following procedure configures a one-way Sync Pipe with a PingDirectory Server as the Sync Source and a generic sync destination. The procedure uses the `create-sync-pipe-config` tool in interactive command-line mode and highlights the differences for configuring a Sync Pipe in notification mode.

Considerations for configuring sync classes

When configuring a Sync Class for a Sync Pipe in notification mode, consider the following:

- Exclude any operational attributes from synchronizing to the destination so that its before and after values are not recorded in the change log. For example, the following attributes can be excluded: `creatorsName`, `createTimeStamp`, `ds-entry-unique-id`, `modifiersName`, and

`modifyTimeStamp`. Filter the changes at the change log level instead of making the changes in the Sync Class to avoid extra configuration settings with the following:

- Use the directory server's `changelog-exclude-attribute` property with (`+`) to exclude all operational attributes (`change-log-exclude-attribute:+`).
- Configure a Sync Class that sets the `excluded-auto-mapped-source-attributes` property to each operational attribute to exclude from the synchronization process.
- Use the directory server's `changelog-exclude-attribute` property to specify each operational attribute to exclude in the synchronization process. Set the configuration using the `dsconfig` tool on the directory server Change Log Backend menu. For example, `setchangelog-exclude-attribute:modifiersName`.

- Use the `destination-create-only-attribute` advanced property on the Sync Class. This property sets the attributes to include on CREATE operations only.
- Use the `replace-all-attr-values` advanced property on the Sync Class. This property specifies whether to use the ADD and DELETE modification types (reversible), or the REPLACE modification type (non-reversible) for modifications to destination entries. If set to `true`, REPLACE is used.
- If targeting specific attributes that require higher performance throughput, consider implementing change log indexing. See *Synchronize through PingDirectoryProxy servers* for more information.

Create the sync pipe

About this task

The initial configuration steps show how to set up a single Sync Pipe from a directory server instance to a generic Sync Destination.

Before starting:

- Place any third-party libraries in the `<server-root>/lib/extensions` folder.
- Implement a server extension for any custom endpoints and place it in the appropriate directory.

Steps

1. If necessary, start PingDataSync Server:

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

2. Run the **`create-sync-pipe-config`** tool.

```
$ bin/create-sync-pipe-config
```

3. At the Initial Synchronization Configuration Tool prompt, press Enter to continue.
4. On the Synchronization Mode menu, select the option for notification mode.
5. On the Synchronization Directory menu, enter the option to create a one-way Sync Pipe in notification mode from directory to a generic client application.

Configure the sync source

About this task

Steps

1. On the Source Endpoint Type menu, enter the option for the Sync Source type.
2. Choose a pre-existing Sync Source, or create a new sync source.
3. Enter a name for the Source Endpoint and a name for the Sync Source.

4. Enter the base DN for the directory server used for LDAP searches, such as `dc=example,dc=com`, and press Enter to return to the menu. If entering more than one base DN, make sure they do not overlap.
5. On the Server Security menu, select the type of communication that PingDataSync Server will use with endpoint servers.
6. Enter the host and port of the first Source Endpoint server. The Sync Source can specify a single server or multiple servers in a replicated topology. PingDataSync Server contacts this first server if it is available, then contacts the next highest priority server if the first server is unavailable. The server tests the connection.
7. On the Sync User Account menu, enter the DN of the sync user account and password, or press Enter to accept the default, `cn=Sync User,cn=Root DNs,cn=config`. This account allows PingDataSync Server to access the source endpoint server.

Configure the destination endpoint server

About this task

Steps

1. On the Destination Endpoint Type menu, select the type of datastore on the endpoint server. In this example, select the option for Custom.
2. Enter a name for the Destination Endpoint and a name for the Sync Destination.
3. On the Notifications Setup menu, select the language (Java or Groovy) used to write the server extension.
4. Enter the fully qualified name of the Server SDK extension that implements the abstract class. A Java, extension should reside in the `/lib/extensions` directory. A Groovy script should reside in the `/lib/groovy-scripted-extensions` directory.
5. Configure any user-defined arguments needed by the server extension. Typically, these are connection arguments, which are defined by the extension itself. The values are then entered here and stored in the server configuration.
6. Configure the maximum number of before and after values for all changed attributes. Notification mode requires this. Set the cap to something well above the maximum number of values that any synchronized attribute will have. If this cap is exceeded, PingDataSync Server issues an alert. For this example, we accept the default value of 200.

```
Enter a value for the max changelog before/after values,
or -1 for no limit [200]:
```

7. Configure any key attributes in the change log that must be included in every notification. These attributes can be used to find the destination entry corresponding to the source entry, and are present regardless of whether the attributes changed. Later, any attributes used in a Sync Class include-filter must also be configured as key attributes in the Sync Class.
8. In both standard and notification modes, the Sync Pipe processes the changes concurrently with multiple threads. If changes must be applied strictly in order, the number of Sync Pipe worker threads will be reduced to 1. This will limit the maximum throughput of the Sync Pipe.

Next steps
The rest of the configuration steps follow the same process as a standard synchronization mode Sync Pipe. See The Sync User account for more information.

Access control filtering on the sync pipe

PingDataSync Server provides an advanced Sync Pipe configuration property, `filter-changes-by-user`, which performs access control filtering on a changelog entry for a specific user.

Since the changelog entry contains data from the target entry, the access controls filter out attributes that the user does not have the privileges to see before it is returned. For example, values in the changes, `ds-changelog-before-values`, `ds-changelog-after-values`, `ds-changelog-entry-key-attr-values`, and `deletedEntryAttrs` are filtered out through access control instructions.

> ⓘ **Note:**
>
> This property is only available for notification mode and can be configured using the **`create-sync-pipe-config`** or the **`dsconfig`** tool.

The source server must be a PingDirectory Server or Nokia 8661 Directory Server, or a PingDirectoryProxy Server or Nokia 8661 Directory Proxy Server that points to a PingDirectory Server or Nokia 8661 Directory Server.

Considerations for access control filtering

- The directory server will not return the changelog entry if the user is not allowed to see the target entry.
- The directory server strips out any attributes that the user is not allowed to see.
- If no changes are left in the entry, no changelog entry will be returned.
- If only some attributes are stripped out, the changelog entry will be returned.
- Access control filtering on a specific attribute value is not supported. Either all attribute values are returned or none.
- If a sensitive attribute policy is used to filter attributes when a client normally accesses the directory server, this policy will not be taken into consideration during notifications since the Sync User is always connecting using the same method. Configure access controls to filter out attributes, not based on the type of connection made to the server, but based on who is accessing the data. The `filter-changes-by-user` property will be able to evaluate if that person should have access to these attributes.

Configure the sync pipe to filter changes by access control instructions

About this task

Steps

1. Set the filter-changes-by-user property to filter changes based on access controls for a specific user.

```
$ bin/dsconfig set-sync-pipe-prop \
  --pipe-name "Notifications Sync Pipe" \
  --set "filter-changes-by-user:uid=admin,dc=example,dc=com"
```

2. On the source directory server, set the `report-excluded-changelog-attributes` property to include the names of users that have been removed through access control filtering. This will allow PingDataSync Server to warn about attributes that were supposed to be synchronized but were filtered out. This step is recommended but not required.

```
$ bin/dsconfig set-backend-prop \
--backend-name "changelog" \
  --set "report-excluded-changelog-attributes:attribute-names"
```

> ⓘ **Note:**
>
> PingDataSync Server only uses the `attribute-names` setting for the PingDirectory Server's `report-excluded-changelog-attributes` property. It does not use the `attribute-counts` setting for the property.

Configuring Synchronization with SCIM

PingDataSync Server provides data synchronization between directory servers or proxy servers and System for Cross-domain Identity Management (SCIM) 1.1 applications over HTTP. Synchronization can be done with custom SCIM applications, or with the PingData PingDirectory Server and PingDirectoryProxy Server configured as SCIM servers using the SCIM extension.

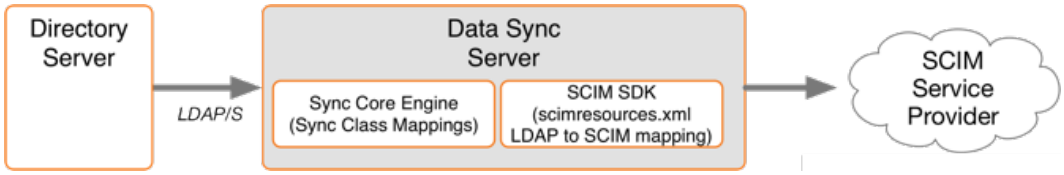
Synchronize with a SCIM sync destination overview

The SCIM 1.1 protocol is designed to make managing user identity in cloud-based applications and services easier. SCIM enables provisioning identities, groups, and passwords to, from, and between clouds. PingDataSync Server can be configured to synchronize with SCIM service providers.

> ⓘ **Note:**
>
> Both the Ping Identity PingDirectory Server and PingDirectoryProxy Server can be configured to be SCIM servers using the SCIM HTTP Servlet Extension.

PingDataSync Server is LDAP-centric and operates on LDAP attributes. The SCIM sync destination server component acts as a translation layer between a SCIM service provider's schema and an LDAP representation of the entries. While PingDataSync Server is LDAP-centric and typically at least one endpoint is an LDAP Directory Server, this is not a strict requirement. For example, a JDBC to SCIM sync pipe can be configured.

PingDataSync Server contains sync classes that define how source and destination entries are correlated. The SCIM sync destination contains its own mapping layer, based on `scimresources.xml` that maps LDAP schema to and from SCIM.



> ⓘ **Note:**
>
> PingDataSync Server can use SCIM only as a sync destination. There is no mechanism in the SCIM protocol for detecting changes, so it cannot be used as a Sync Source.

SCIM destination configuration objects

The `SCIMSyncDestination` object defines a SCIM 1.1 service provider Sync Pipe destination that is accessible over HTTP through the SCIM protocol. It is configured with the following properties:

- `server` – Specifies the names of the SCIM External Servers that are used as the destination of synchronization.
- `resource-mapping-file` – Specifies the path to the `scim-resources.xml` file, a configuration file that defines the SCIM schema and maps it to the LDAP schema. This file is located in `<server root>/config/scim-resources.xml` by default. This file can be customized to define and expose deployment-specific resources.
- `rename-policy` – Specifies how to handle the rename of a SCIM resource.

The SCIM Sync Destination object is based on the SCIM SDK. Before configuring a SCIM destination, review the following documents on the Simple Cloud web site:

- SCIM Core Schema
- SCIM REST API

Considerations for synchronizing to a SCIM destination

When configuring an LDAP to SCIM Sync Pipe, consider the following:

Use `scim-resources.xml` for attribute and DN mappings

There are two layers of mapping: once at the Sync Class level and again at the SCIM Sync Destination level in the `scim-resources.xml` file. To reduce complexity, do all possible mappings in the `scim-resources.xml` file.

Avoid groups unless the SCIM ID is DN based

Group synchronization is supported if the SCIM ID is based on the DN. If the SCIM ID is not the DN itself, it must be one of the components of the RDN, meaning that the DNs of group members must contain the necessary attribute.

SCIM modifies entries using PUT

The SCIM Sync Destination modifies entries using the full HTTP PUT method. For every modify, SCIM replaces the entire resource with the updated resource. For information about the implications of this on password updates, see *Password considerations with SCIM*.

Rename a SCIM resource

The SCIM 1.1 protocol does not support changes that require the SCIM resource to be renamed, such as a MODDN operation. Instead, when a change is detected to an attribute value that is used as part of the SCIM ID attribute, PingDataSync Server handles it in one of the following ways:

- Deletes the specified SCIM resource and then adds the new resource with the new SCIM ID.
- Adds the new resource with the new SCIM ID and then deletes the old resource.
- Skips the rename portion of the change. If renames are expected on the source endpoint, a careful set of destination-correlation attributes should be chosen so that the destination can still be found after it is renamed on the source.

Configure this by setting the `rename-policy` property of the SCIM sync destination.

Password considerations with SCIM

Because the SCIM sync destination modifies entries using a full PUT method, special considerations need to be made for password attributes.

A Ping Identity SCIM Server allows password attributes to be omitted from a change when they have not been modified by an operation. This prevents passwords from inadvertently being overwritten during the PUT operation, which does not include the password attribute. Ideally, other SCIM service providers will not wipe a password because a PUT request does not contain it. Check with the SCIM vendor to confirm this behavior before starting a SCIM sync pipe.

Configure synchronization with SCIM

Configure synchronization with SCIM using the **`create-sync-pipe-config`** utility and the **`dsconfig`** command. Configuring synchronization between an LDAP server and a SCIM service provider includes the following:

- Configure one external server for every physical endpoint.
- Configure the Sync Source server and designate the external servers that correspond to the source server.
- Configure the Sync Destination server and designate the external servers that correspond to the SCIM sync destination.
- Configure the LDAP to SCIM Sync Pipe.
- Configure the Sync Classes. Each Sync Class represents a type of entry that needs to be synchronized. When specifying a Sync Class for synchronization with a SCIM service provider,

avoid including attribute and DN mappings. Instead use the Sync Class to specify the operations to synchronize and which correlation attributes to use.
- Set the evaluation order for the Sync Classes to define the processing precedence for each class.
- Configure the `scim-resources.xml` file. If possible, change the `<resourceIDMapping>` element(s) to use whatever the SCIM Service Provider uses as the SCIM ID.
- Set Up Communication for each External Server. Run **`prepare-endpoint-server`** once for every LDAP external server that is part of the Sync Source.
- Use **`realtime-sync`** to start the Sync Pipe.

Configure the external servers

About this task

Perform the following to configure an external server for each host in the deployment:

Steps

1. Configure a PingDirectory Server as an external server, which will later be configured as a Sync Source. On PingDataSync Server, run the following **`dsconfig`** command:

```
$ bin/dsconfig create-external-server \
  --server-name source-ds \
  --type ping-identity-ds \
  --set server-host-name:ds1.example.com \
  --set server-port:636 \
  --set "bind-dn:cn=Directory Manager" \
  --set password:secret \
  --set connection-security:ssl \
  --set key-manager-provider:Null \
  --set trust-manager-provider:JKS
```

2. Configure the SCIM server as an external server, which will later be configured as a Sync Destination. The `scim-service-url` property specifies the complete URL used to access the SCIM service provider. The user-name property specifies the account used to connect to the SCIM service provider. By default, the value is `cn=Sync User,cn=Root DNs,cn=config`. Some SCIM service providers may not have the user name in DN format.

```
$ bin/dsconfig create-external-server \
  --server-name scim \
  --type scim \
  --set scim-service-url:https://scim1.example.com:8443 \
  --set "user-name:cn=Sync User,cn=Root DNs,cn=config" \
  --set password:secret \
  --set connection-security:ssl \
  --set hostname-verification-method:strict \
  --set trust-manager-provider:JKS
```

Configure the PingDirectory Server sync source

About this task

Configure the Sync source for the synchronization network. More than one external server can be configured to act as the Sync source for failover purposes. If the source is a PingDirectory Server, also configure the following items:

- Enable the changelog password encryption plugin on any directory server that will receive password modifications. This plugin intercepts password modifications, encrypts the password, and adds an encrypted attribute to the change log entry.

- Configure the `changelog-deleted-entry-include-attribute` property on the changelog backend, so that PingDataSync Server can record which attributes were removed during a DELETE operation.

Perform the following steps to configure the Sync source:

Steps

1. Run **`dsconfig`** to configure the external server as the Sync source. Based on the previous example where the PingDirectory Server was configured as `source-ds`, run the following command:

```
$ bin/dsconfig create-sync-source --source-name source \
  --type ping-identity \
  --set base-dn:dc=example,dc=com \
  --set server:source-ds \
  --set use-changelog-batch-request:true
```

2. Enable the change log password encryption plugin on any server that receives password modifications. The encryption key can be copied from the output, if displayed, or accessed from the `<server-root>/bin/sync-pipe-cfg.txt` file, if the **`create-sync-pipe-config`** tool was used to create the sync pipe.

```
$ bin/dsconfig set-plugin-prop \
  --plugin-name "Changelog Password Encryption" \
  --set enabled:true \
  --set changelog-password-encryption-key:<key>
```

3. On PingDataSync Server, set the decryption key used to decrypt the user password value in the change log entries. The key allows the user password to be synchronized to other servers that do not use the same password storage scheme.

```
$ bin/dsconfig set-global-sync-configuration-prop \
  --set changelog-password-decryption-key:ej5u9e39pq-68
```

4. Configure the `changelog-deleted-entry-include-attribute` property on the changelog backend.

```
$ bin/dsconfig set-backend-prop --backend-name changelog \
  --set changelog-deleted-entry-include-attribute:objectClass
```

Configure the SCIM sync destination

Configure the SCIM sync destination to synchronize data with a SCIM service provider. Run the `dsconfig` command:

```
$ bin/dsconfig create-sync-destination \
  --destination-name scim \
  --type scim \
  --set server:scim
```

Configure the sync pipe, sync classes, and evaluation order

About this task

Configure a Sync Pipe for LDAP to SCIM synchronization, create Sync classes for the Sync Pipe, and set the evaluation order index for the Sync classes.

ⓘ **Note:**

The Synchronization mode must be set to Standard. Notification Mode cannot be used with SCIM.

Steps

1. Once the source and destination endpoints are configured, configure the Sync Pipe for LDAP to SCIM synchronization. Run the **`dsconfig`** command to configure an LDAP-to-SCIM Sync Pipe:

```
$ bin/dsconfig create-sync-pipe \
  --pipe-name ldap-to-scim \
  --set sync-source:source \
  --set sync-destination:scim
```

2. The next set of steps define three Sync Classes. The first Sync Class is used to match user entries in the Sync Source. The second class is used to match group entries. The third class is a DEFAULT class that is used to match all other entries.

   Run the `dsconfig` command to create the first Sync Class and set the Sync Pipe Name and Sync Class name:

```
$ bin/dsconfig create-sync-class \
  --pipe-name ldap-to-scim \
  --class-name user
```

3. Use `dsconfig` to set the base DN and filter for this Sync class. The `include-base-dn` property specifies the base DN in the source, which is `ou=people,dc=example,dc=com` by default. This Sync Class is invoked only for changes at the `ou=people` level. The include-filter property specifies an LDAP filter that tells PingDataSync Server to include `inetOrgPerson` entries as user entries. The `destination-correlation-attributes` specifies LDAP attributes that allow PingDataSync Server to find the destination resource on the SCIM server. The value of this property will vary. See *Identify a SCIM resource at the destination* for details.

```
$ bin/dsconfig set-sync-class-prop \
  --pipe-name ldap-to-scim \
  --class-name user \
  --add include-base-dn:ou=people,dc=example,dc=com \
  --add "include-filter:(objectClass=inetOrgPerson)" \
  --set destination-correlation-attributes:externalId
```

4. Create a second Sync class, which is used to match group entries:

```
$ bin/dsconfig create-sync-class \
  --pipe-name ldap-to-scim \
  --class-name group
```

5. For the second Sync class, set the base DN and the filters to match the group entries.

```
$ bin/dsconfig set-sync-class-prop \
  --pipe-name ldap-to-scim \
  --class-name group \
  --add include-base-dn:ou=groups,dc=example,dc=com \
  --add "include-filter:(|(objectClass=groupOfEntries)\
    (objectClass=groupOfNames)(objectClass=groupOfUniqueNames)\
    (objectClass=groupOfURLs))"
```

6. For the third Sync class, create a DEFAULT Sync class that is used to match all other entries. To synchronize changes from only user and group entries, set `synchronize-creates`, `synchronize-modifies`, and `synchronize-delete` to false.

```
$ bin/dsconfig create-sync-class \
  --pipe-name ldap-to-scim \
```

```
  --class-name DEFAULT \
  --set evaluation-order-index:99999 \
  --set synchronize-creates:false \
  --set synchronize-modifies:false \
  --set synchronize-deletes:false
```

7. After all of the Sync classes needed by the Sync Pipe are configured, set the evaluation order index for each Sync class. Classes with a lower number are evaluated first. Run **`dsconfig`** to set the evaluation order index for the Sync class. The actual number depends on the deployment.

```
$ bin/dsconfig set-sync-class-prop \
  --pipe-name ldap-to-scim \
  --class-name user \
  --set evaluation-order-index:100
```

Configure communication with the source server

Configure communication between PingDataSync Server and the LDAP source servers with the `prepare-endpoint-server` tool. If user accounts do not exist, this tool creates the appropriate user account and its privileges. Also, because the source is an PingDirectory Server, this tool enables the changelog.

> ⓘ **Note:**
>
> The **`prepare-endpoint-server`** tool can only be used on LDAP directory servers. For the SCIM Server, manually create a sync user entry.
>
> Run the **`prepare-endpoint-server`** command to setup communication between PingDataSync Server and the source server(s). The tool will prompt for the bind DN and password to create the user account and enable the change log.

```
$ bin/prepare-endpoint-server \
  --hostname ds1.example.com \
  --port 636 \
  --useSSL \
  --trustAll \
  --syncServerBindDN "cn=Sync User,cn=Root DNs,cn=config" \
  --syncServerBindPassword "password" \
  --baseDN "dc=example,dc=com" \
  --isSource
```

Start the sync pipe

About this task
The **`realtime-sync`** tool sets a specific starting point for real-time synchronization, so that changes made before the current time are ignored.

Steps

1. Run the **`realtime-sync`** tool to set the startpoint for the Sync source.

```
$ bin/realtime-sync set-startpoint \
  --end-of-changelog \
  --pipe-name ldap-to-scim
```

2. When ready to start synchronization, run the following command:

```
$ bin/realtime-sync start \
  --pipe-name ldap-to-scim \
```

```
  --no-prompt
```

Map LDAP schema to SCIM resource schema

The resources configuration file is used to define the SCIM resource schema and its mapping to LDAP schema. The default configuration of the `scim-resources.xml` file provides definitions for standard SCIM Users and Groups resources, and mappings to standard LDAP `inetOrgPerson` and `groupOfUniqueNames` object classes. It is installed with the PingDirectory Server. This file can be customized by adding extension attributes to the Users and Groups resources, or by adding new extension resources. The resources file is composed of a single `<resources>` element, containing one or more `<resource>` elements.

The default configuration maps the SCIM resource ID to the LDAP `entryUUID` attribute. In all cases, this must be changed to match the attribute that the destination SCIM service provider is using for its SCIM resource ID. For example, if the destination uses the value of the `uid` attribute, modify the `scim-resources.xml` file to change the `resourceIDMapping` as follows:

```
<resourceIDMapping ldapAttribute="uid"/>
```

Ideally, this would be an attribute that exists on the source LDAP entry. If not, PingDataSync Server can construct it using a Constructed Attribute Mapping. For example, the SCIM service provider used the first and last initials of the user, concatenated with the employee ID (given by the `eid`attribute)as the SCIM resource ID. In this case, an attribute mapping would be constructed as follows:

```
$ dsconfig create-attribute-mapping \
  --map-name MyAttrMap \
  --mapping-name scimID \
  --type constructed \
  --set 'value-pattern:{givenname:/^(.)(.*)/$1/s}{sn:/^(.)(.*)/$1/s}{eid}'
```

This creates an attribute called `scimID` on the mapped entry when processed by the Sync engine. For example, if the user's name was John Smith, with employee ID 12345, then the `scimID` would be `js12345`. Once this is done, configure the `scim-resources.xml` file as follows:

```
<resourceIDMapping ldapAttribute="scimID" />
```

This will cause it to pull out the constructed `scimID`value from the entry and use that at the SCIM resource ID when making requests to the service provider.

> ⓘ **Note:**
>
> Constructed attribute mappings support multivalued source attributes for conditional (using the `conditional-value-pattern` property) and non-conditional (using the `value-pattern` property) value patterns. Only one of the source attributes that contribute to a given value pattern can be multivalued.

For any given SCIM resource endpoint, only one `<LDAPAdd>` template can be defined, and only one `<LDAPSearch>` element can be referenced. If entries of the same object class can be located under different subtrees or base DNs of the PingDirectory Server, then a distinct SCIM resource must be defined for each unique entry location in the Directory Information Tree. If using the SCIM HTTP Servlet Extension for the PingDirectory Server, this can be implemented in many ways, such as:

- Create multiple SCIM servlets, each with a unique `resources.xml` configuration, and each running under a unique HTTP connection handler.
- Create multiple SCIM servlets, each with a unique `resources.xml` configuration, each running under a single, shared HTTP connection handler, but each with a unique context path.

LDAP attributes are allowed to contain characters that are invalid in XML (because not all valid UTF-8 characters are valid XML characters). Make sure that any attributes that contain binary data are declared using `dataType=binary` in the `scim-resources.xml` file. When using the Identity Access API, make sure that the underlying LDAP schema uses the Binary or Octet String attribute syntax for attributes that contain binary data. This instructs the server to base64-encode the data before returning it to clients.

If attributes that are not declared as binary in the schema and contain binary data (or just data that is invalid in XML), the server will check for this before returning them to the client. If the client has set the content-type to XML, then the server may choose to base64-encode any values that include invalid XML characters. When this is done, a special attribute is added to the XML element to alert the client that the value is base64-encoded. For example:

```
<scim:value base64Encoded="true">AAABPB0EBZc=</scim:value>
```

The remainder of this section describes the mapping elements available in the `scimresources.xml` file.

<resource> element

A `resource` element has the following XML attributes:

- `schema`: a required attribute specifying the SCIM schema URN for the resource. Standard SCIM resources already have URNs assigned for them, such as `urn:scim:schemas:core:1.0`. A new URN must be obtained for custom resources using any of the standard URN assignment methods.
- `name`: a required attribute specifying the name of the resource used to access it through the SCIM REST API.
- `mapping`: a custom Java class that provides the logic for the resource mapper. This class must extend the `com.unboundid.scim.ldap.ResourceMapper` class.

A `resource` element contains the following XML elements in sequence:

- `description`: a required element describing the resource.
- `endpoint`: a required element specifying the endpoint to access the resource using the SCIM REST API.
- `LDAPSearchRef`: a mandatory element that points to an `LDAPSearch` element. The `LDAPSearch`element allows a SCIM query for the resource to be handled by an LDAP service and also specifies how the SCIM resource ID is mapped to the LDAP server.
- `LDAPAdd`: an optional element specifying information to allow a new SCIM resource to be added through an LDAP service. If the element is not provided then new resources cannot be created through the SCIM service.
- `attribute`: one or more elements specifying the SCIM attributes for the resource.

<attribute> element

An `attribute` element has the following XML attributes:

- `schema`: a required attribute specifying the schema URN for the SCIM attribute. If omitted, the schema URN is assumed to be the same as that of the enclosing resource, so this only needs to be provided for SCIM extension attributes. Standard SCIM attributes already have URNs assigned for them, such as `urn:scim:schemas:core:1.0`. A new URN must be obtained for custom SCIM attributes using any of the standard URN assignment methods.
- `name`: a required attribute specifying the name of the SCIM attribute.
- `readOnly`: an optional attribute indicating whether the SCIM sub-attribute is not allowed to be updated by the SCIM service consumer. The default value is `false`.
- `required`: an optional attribute indicating whether the SCIM attribute is required to be present in the resource. The default value is `false`.

An `attribute` element contains the following XML elements in sequence:

- `description`: a required element describing the attribute. Then just one of the following elements:
- `simple`: specifies a simple, singular SCIM attribute.

- `complex`: specifies a complex, singular SCIM attribute.
- `simpleMultiValued`: specifies a simple, multi-valued SCIM attribute.
- `complexMultiValued`: specifies a complex, multi-valued SCIM attribute.

<simple> element

A `simple` element has the following XML attributes:

- `dataType`: a required attribute specifying the simple data type for the SCIM attribute. The following values are permitted: `binary`, `boolean`, `dateTime`, `decimal`, `integer`, and `string`.
- `caseExact`: an optional attribute that is only applicable for string data types. It indicates whether comparisons between two string values use a case-exact match or a case- ignore match. The default value is `false`.

A `simple` element contains the following XML element in sequence:

- `mapping`: an optional element specifying a mapping between the SCIM attribute and an LDAP attribute. If this element is omitted, the SCIM attribute has no mapping and the SCIM service ignores any values provided for the SCIM attribute.

<complex> element

The `complex` element does not have any XML attributes. It contains the following XML element:

- `subAttribute`: one or more elements specifying the sub-attributes of the complex SCIM attribute, and an optional mapping to LDAP. The standard type, primary, and display sub-attributes do not need to be specified.

<simpleMultiValued> element

A `simpleMultiValued` element has the following XML attributes:

- `childName`: a required attribute specifying the name of the tag that is used to encode values of the SCIM attribute in XML in the REST API protocol. For example, the tag for the standard emails SCIM attribute is email.
- `dataType`: a required attribute specifying the simple data type for the plural SCIM attribute (i.e. the data type for the value sub-attribute). The following values are permitted: `binary`, `boolean`, `dateTime`, `integer`, and `string`.
- `caseExact`: an optional attribute that is only applicable for string data types. It indicates whether comparisons between two string values use a case-exact match or a case- ignore match. The default value is `false`.

A `simpleMultiValued` element contains the following XML elements in sequence:

- `canonicalValue`: specifies the values of the type sub-attribute that is used to label each individual value, and an optional mapping to LDAP.
- `mapping`: an optional element specifying a default mapping between the SCIM attribute and an LDAP attribute.

<complexMultiValued> element

A `complexMultiValued` element has the following XML attributes:

- `tag`: a required attribute specifying the name of the tag that is used to encode values of the SCIM attribute in XML in the REST API protocol. For example, the tag for the standard `addresses` SCIM attribute is `address`.

A `complexMultiValued` element contains the following XML elements in sequence:

- `subAttribute`: one or more elements specifying the sub-attributes of the complex SCIM attribute. The standard `type`, `primary`, and `display` sub-attributes do not need to be specified.

`canonicalValue`: specifies the values of the type sub-attribute that is used to label each individual value, and an optional mapping to LDAP.

<subAttribute> element

A `subAttribute` element has the following XML attributes:

- `name`: a required element specifying the name of the sub-attribute.
- `readOnly`: an optional attribute indicating whether the SCIM sub-attribute is not allowed to be updated by the SCIM service consumer. The default value is `false`.
- `required`: an optional attribute indicating whether the SCIM sub-attribute is required to be present in the SCIM attribute. The default value is `false`.
- `dataType`: a required attribute specifying the simple data type for the SCIM sub-attribute. The following values are permitted: `binary`, `boolean`, `dateTime`, `integer`, and `string`.
- caseExact:an optional attribute that is only applicable for string data types. It indicates whether comparisons between two string values use a case-exact match or a case- ignore match. The default value is `false`.

A `subAttribute` element contains the following XML elements in sequence:

- `description`: a required element describing the sub-attribute.
- `mapping`: an optional element specifying a mapping between the SCIM sub-attribute and an LDAP attribute. This element is not applicable within the `complexMultiValued` element.

<canonicalValue> element

A `canonicalValue` element has the following XML attributes:

- `name`: specifies the value of the type sub-attribute. For example, work is the value for emails, phone numbers and addresses intended for business purposes.

A `canonicalValue` element contains the following XML element in sequence:

- `subMapping`: an optional element specifying mappings for one or more of the sub-attributes. Any sub-attributes that have no mappings will be ignored by the mapping service.

<mapping> element

A `mapping` element has the following XML attributes:

- `ldapAttribute`: a required element specifying the name of the LDAP attribute to which the SCIM attribute or sub-attribute map.
- `transform`: an optional element specifying a transformation to apply when mapping an attribute value from SCIM to LDAP, and LDAP to SCIM. The available transformations are described in *Map LDAP schema to SCIM resource schema*.

<subMapping> element

A `subMapping` element has the following XML attributes:

- `name`: a required element specifying the name of the sub-attribute that is mapped.
- `ldapAttribute`: a required element specifying the name of the LDAP attribute to which the SCIM sub-attribute maps.
- `transform`: an optional element specifying a transformation to apply when mapping an attribute value from SCIM to LDAP and vice-versa. The available transformations are described later. Available transformations are described in *Map LDAP schema to SCIM resource schema*.

<LDAPSearch> element

A `LDAPSearch` element has the following XML attributes:

- `baseDN`: a required element specifying the LDAP search base DN to be used when querying for the SCIM resource.

- `filter`: a required element specifying an LDAP filter that matches entries representing the SCIM resource. This filter is typically an equality filter on the LDAP object class.
- `resourceIDMapping`: an optional element specifying a mapping from the SCIM resource ID to an LDAP attribute. When the element is omitted, the resource ID maps to the LDAP entry DN.

> ⓘ **Note:**
>
> The `LDAPSearch` element can be added as a top-level element outside of any `<Resource>` elements, and then referenced within them with an `ID` attribute.

<resourceIDMapping> element

A `resourceIDMapping` element has the following XML attributes:

- `ldapAttribute`: a required element specifying the name of the LDAP attribute to which the SCIM resource ID maps.
- `createdBy`: a required element specifying the source of the resource ID value when a new resource is created by the SCIM consumer using a POST operation. Allowable values for this element include `<scim-consumer>`, meaning that a value must be present in the initial resource content provided by the SCIM consumer, or directory, (as would be the case if the mapped LDAP attribute is `entryUUID`).

If the LDAP attribute value is not listed as destination correlation attribute, this setting is not used by PingDataSync Server.

The following example illustrates an `LDAPSearch` element that contains a `resourceIDMapping` element:

```
<LDAPSearch id="userSearchParams">
  <baseDN>ou=people,dc=example,dc=com</baseDN>
  <filter>(objectClass=inetOrgPerson)</filter>
  <resourceIDMapping ldapAttribute="entryUUID" createdBy="directory"/>
</LDAPSearch>
```

<LDAPAdd> element

A `LDAPAdd` element has the following XML attributes:

- `DNTemplate`: a required element specifying a template that is used to construct the DN of an entry representing a SCIM resource when it is created. The template may reference values of the entry after it has been mapped using `{ldapAttr}`, where `ldapAttr` is the name of an LDAP attribute.
- `fixedAttribute`: zero or more elements specifying fixed LDAP values to be inserted into the entry after it has been mapped from the SCIM resource.

<fixedAttribute> element

A `fixedAttribute` element has the following XML attributes:

- `ldapAttribute`: a required attribute specifying the name of the LDAP attribute for the fixed values.
- `onConflict`: an optional attribute specifying the behavior when the LDAP entry already contains the specified LDAP attribute. The default value `merge` indicates that the fixed values should be merged with the existing values. The value `overwrite` indicates that the existing values are to be overwritten by the fixed values. The value `preserve` indicates that no changes should be made.

A `fixedAttribute`element contains the following XML element:

- `fixedValue`: one or more elements specifying the fixed LDAP values.

Identify a SCIM resource at the destination

When a SCIM Sync Destination needs to synchronize a change to a SCIM resource on the destination SCIM server, it must first fetch the destination resource. If the destination resource ID is known, the resource will be retrieved by its ID. If not, a search is performed using the mapped destination correlation

attributes. Configuring this requires coordination between the Sync Class and the `scim-resources.xml` mapping file.

The `scim-resources.xml` mapping file treats the value of the `<resourceIDMapping>` element's `ldapAttribute` attribute as the SCIM ID of the source entry. If this value is also listed as a value of the Sync Class's `destination-correlation-attributes` property, then the value of this LDAP attribute is used as the SCIM ID of the destination resource.

If no value of `destination-correlation-attributes` matches the `<resourceIDMapping>` element's `ldapAttribute` attribute, the SCIM ID of the destination resource is considered unknown. In this case, the SCIM Sync Destination treats the values of `destination- correlation-attributes` as search terms, using them to construct a filter for finding the destination resource. Each value of `destination-correlation-attributes` will be mapped to a corresponding SCIM attribute name, and equality matches will be used in the resulting filter.

If the `ldapAttribute` value is not listed as a destination correlation attribute, this setting is not used by PingDataSync Server.

The following table illustrates an `LDAPSearch` element that contains a `resourceIDMapping` element:

Identifying a SCIM resource

Method for retrieving SCIM resource	Condition	Example condition	Example request
Retrieve resource directly	Used if a `destination-correlation-attribute` value matches the `<resourceIDMapping>ldapAttribute` value.	`destination-correlation-attribute=mail,uid;<resourceIDMapping ldapAttribute="mail" createdBy= "directory"/>`	`GET scim/Users/ person@example.com`
Retrieve resource using search	Used if no `destination-correlation-attribute`value matches the `<resourceIDMapping>ldapAttribute` value.	`destination-correlation-attribute=mail,uid;<resourceIDMapping ldapAttribute="entryUUID" createdBy = "directory"/>`	`GET /scim/Users? filter=emails+eq +"person@example.com" and userName+eq"person"`

The unique ID of a destination SCIM resource will most likely be unknown, and the search method will need to be used. However, not all SCIM service providers support the use of filters. Therefore, not all SCIM service providers may be usable as SCIM Sync Destinations.

Managing Logging, Alerts, and Alarms

Each PingData server supports extensive logging features to track all aspects of the PingData topology.

Logs and log publishers

PingData servers support different types of log publishers that can be used to provide the monitoring information for operations, access, debug, and error messages that occur during normal server processing.

The server provides a standard set of default log files as well as mechanisms to configure custom log publishers with their own log rotation and retention policies.

Types of log publishers

Several types of log publishers can be used to log processing information about the server, including:

Audit loggers

Provide information about actions that occur within the server. Specifically, this type of log records all changes applied, detected or failed; dropped operations that were not completed; changes dropped due to being out of scope, or no changes needed for an operation. The log also shows the entries that were involved in a process.

Error loggers

Provide information about warnings, errors, or significant events that occur within the server.

Debug loggers

Provide detailed information about processing performed by the server, including any exceptions caught during processing, detailed information about data read from or written to clients, and accesses to the underlying database.

Access loggers

Provide information about LDAP operations processed within the server. This log only applies to operations performed in the server. This includes configuration changes, searches of monitor data, and bind operations for authenticating administrators using the command-line tools and the Administrative Console.

View the list of log publishers

View the list of log publishers on each server using the **dsconfig** tool:

```
$ bin/dsconfig list-log-publishers
Log Publisher                       : Type              : enabled
------------------------------------:-------------------:--------
Debug ACI Logger                    : debug-access      : false
Expensive Operations Access Logger : file-based-access : false
Failed Operations Access Logger
File-Based Access Logger
File-Based Audit Logger
: file-based-access : true
: file-based-access : true
: file-based-audit  : false
File-Based Debug Logger
File-Based Error Logger
Replication Repair Logger
: file-based-debug  : false
: file-based-error  : true
: file-based-error  : true
```

Log compression

PingData servers support the ability to compress log files as they are written. Because of the inherent problems with mixing compressed and uncompressed data, compression can only be enabled when the logger is created. Compression cannot be turned on or off once the logger is configured. If the server encounters an existing log file at startup, it will rotate that file and begin a new one rather than attempting to append it to the previous file.

Compression is performed using the standard **gzip** algorithm. Because it can be useful to have an amount of uncompressed log data for troubleshooting, having a second logger defined that does not use compression may be desired.

Configure compression by setting the `compression-mechanism` property to have the value of **`gzip`** when creating a new logger. See *Creating a new log publisher* on page 1474 for details.

Configuring log file encryption

About this task

The server supports the ability to encrypt log files as they are written. The encrypt-log configuration property controls whether encryption will be enabled for the logger. Enabling encryption causes the log file to have an `.encrypted` extension (and if both encryption and compression are enabled, the extension will be `.gz.encrypted`). Any change that affects the name used for the log file could prevent older files from getting properly cleaned up.

Like compression, encryption can only be enabled when the logger is created. Encryption cannot be turned on or off once the logger is configured. For any log file that is encrypted, enabling compression is also recommended to reduce the amount of data that needs to be encrypted. This will also reduce the overall size of the log file. The **`encrypt-file`** tool (or custom code, using the LDAP SDK's `com.unboundid.util.PassphraseEncryptedInputStream`) is used to access the encrypted data.

To enable encryption, at least one encryption settings definition must be defined in the server. Use the one created during setup, or create a new one with the **`encryption-settings create`** command. By default, the encryption will be performed with the server's preferred encryption settings definition. To explicitly specify which definition should be used for the encryption, the `encryption-settings-definition-id` property can be set with the ID of that definition. It is recommended that the encryption settings definition is created from a passphrase so that the file can be decrypted by providing that passphrase, even if the original encryption settings definition is no longer available. A randomly generated encryption settings definition can also be created, but the log file can only be decrypted using a server instance that has that encryption settings definition.

When using encrypted logging, a small amount of data may remain in an in-memory buffer until the log file is closed. The encryption is performed using a block cipher, and it cannot write an incomplete block of data until the file is closed. This is not an issue for any log file that is not being actively written. To examine the contents of a log file that is being actively written, use the **`rotate-log`** tool to force the file to be rotated before attempting to examine it.

The following commands can be used to set log file encryption:

Steps

1. Use **`dsconfig`** to enable encryption for a Log Publisher. In this example, the File-basedAccess Log Publisher "Encrypted Access" is created, compression is set, and rotation and retention policies are set.

```
$ bin/dsconfig create-log-publisher-prop --publisher-name "Encrypted
Access" \
  --type file-based-access \
  --set enabled:true \
  --set compression-mechanism:gzip \
  --set encryption-settings-definition-
id:332C846EF0DCD1D5187C1592E4C74CAD33FC1E5FC20B726CD301CDD2B3FFBC2B \
  --set encrypt-log:true \
  --set log-file:logs/encrypted-access \
  --set "rotation-policy:24 Hours Time Limit Rotation Policy" \
  --set "rotation-policy:Size Limit Rotation Policy" \
   --set "retention-policy:File Count Retention Policy" \
  --set "retention-policy:Free Disk Space Retention Policy" \
   --set "retention-policy:Size Limit Retention Policy"
```

2. To decrypt and decompress the file:

```
$ bin/encrypt-file --decrypt \
  --decompress-input \
```

```
  --input-file logs/encrypted-access.20180216040332Z.gz.encrypted \
  --output-file decrypted-access
Initializing the server's encryption framework...DoneWriting decrypted
data to file '/ds/PingDirectory/decrypted-access' using a key generated
from encryption settings definition
'332c846ef0dcd1d5187c1592e4c74cad33fc1e5fc20b726cd301cdd2b3ffbc2b'Success
fully wrote 123,456,789 bytes of decrypted data
```

Synchronization logs and messages

TPingDataSync Server provides a standard set of default log files to monitor the server activity. View this set of logs in the `<server-root>/logs` directory. The following default log files are available.

PingDataSync Server logs

Log file	Description
access	File-based Access Log that records LDAP operations processed by PingDataSync Server. Access log records can be used to provide information about problems during operation processing and provide information about the time required to process each operation.
config-audit.log	Records information about changes made to the server configuration in a format that can be replayed using the **`dsconfig`** tool.
errors	File-based Error Log that provides information about warnings, errors, and significant events that are not errors but occur during server processing.
server.out	Records anything written to standard output or standard error, which includes startup messages. If garbage collection debugging is enabled, then the information will be written to `server.out`.
server.pid	Stores the server's processID.
server.status	Stores the timestamp, a status code, and an optional message that provides additional information about the server status.
setup.log	Records messages that occur during the initial server configuration with the **`setup`** command.
sync	File-based Sync Log that records synchronization operations processed by the server. Specifically, the log records all changes applied, detected or failed; dropped operations that were not synchronized; changes dropped due to being out of scope, or no changes needed for synchronization.
sync-pipe-cfg.txt	Records the configuration changes used with the `bin/create-sync-pipe-config` tool. The file is placed wherever the tool is run. Typically, this is in `<server-root>` or in the `bin` directory.
tools	Holds logs for long running utilities. Current and previous copies of the log are present in the directory.
update.log	Records messages that occur during a server upgrade.

Sync log message types

PingDataSync Server logs certain types of log messages with the sync log. Message types can be included or excluded from the logger, or added to a custom log publisher.

Sync log message types

Message type	Description
change-applied	Default summary message. Logged each time a change is applied successfully.
change-detected	Default summary message. Logged each time a change is detected.
change-failed-detailed	Default detail message. Logged when a change cannot be applied. It includes the reason for the failure and details about the change that can be used to manually repair the failure.
dropped-op-type-not-synchronized	Default summary message. Logged when a change is dropped because the operation type (for example, ADD) is not synchronized for the matching Sync Class.
dropped-out-of-scope	Default summary message. Logged when a change is dropped because it does not match any Sync Class.
no-change-needed	Default summary message. Logged each time a change is dropped because the modified source entry is already synchronized with the destination entry.
change-detected-detailed	Optional detail message. Logged each time a change is detected. It includes attribute values for added and modified entries. This information is useful for diagnosing problems, but it causes log files to grow faster, which impacts performance.
entry-mapping-details	Optional detail message. Logged each time a source entry (attributes and DN) are mapped to a destination entry. This information is useful for diagnosing problems, but it causes log files to grow faster, which impacts performance.
change-applied-detailed	Optional detail message. Logged each time a change is applied. It includes attribute values for added and modified entries. This information is useful for diagnosing problems, but it causes log files to grow faster, which impacts performance.
change-failed	Optional summary message. Logged when a change cannot be applied. It includes the reason for the failure but not enough information to manually repair the failure.
intermediate-failure	Optional summary message. Logged each time an attempt to apply a change fails. Note that a subsequent retry of applying the change might succeed.

Creating a new log publisher

About this task

PingData servers provide customization options to create log publishers with the `dsconfig` command. After creating a new log publisher, configure the log retention and rotation policies. For more information, see *Configure log retention and log rotation policies*.

Steps

1. Use the **dsconfig** command to create and configure the new log publisher. (If using **dsconfig** in interactive mode, log publishers are created and managed under the Log Publisher menu.) The following example shows how to create a logger that only logs disconnect operations.

```
$ bin/dsconfig create-log-publisher \
  --type file-based-access --publisher-name "Disconnect Logger" \
  --set enabled:true \
  --set "rotation-policy:24 Hours Time Limit Rotation Policy" \
  --set "rotation-policy:Size Limit Rotation Policy" \
  --set "retention-policy:File Count Retention Policy" \
  --set log-connects:false \
  --set log-requests:false --set log-results:false \
  --set log-file:logs/disconnect.log
```

   To configure compression on the logger, add the following option to the previous command:

```
--set compression-mechanism: gzip
```

   Compression cannot be disabled or turned off once configured for the logger. Determine logging requirements before configuring this option.

2. View log publishers with the following command:

```
$ bin/dsconfig list-log-publishers
```

Configuring log signing

About this task

PingData servers support the ability to cryptographically sign a log to ensure that it has not been modified. For example, financial institutions require tamper-proof audit logs files to ensure that transactions can be properly validated and ensure that they have not been modified by a third-party entity or internally by an unauthorized person.

When enabling signing for a logger that already exists, the first log file will not be completely verifiable because it still contains unsigned content from before signing was enabled. Only log files whose entire content was written with signing enabled will be considered completely valid. For the same reason, if a log file is still open for writing, then signature validation will not indicate that the log is completely valid because the log will not include the necessary "end signed content" indicator at the end of the file.

To validate log file signatures, use the **validate-file-signature** tool provided in the `bin` directory of the server (or the `bat` directory on Windows systems). Once this property is enabled, disable and then re-enable the log publisher for the changes to take effect.

Perform the following steps to configure log signing:

Steps

1. Use **dsconfig** to enable log signing for a Log Publisher. In this example, set the `sign-log` property on the File-based Audit Log Publisher.

```
$ bin/dsconfig set-log-publisher-prop \
  --publisher-name "File-Based Audit Logger" \
  --set sign-log:true
```

2. Disable and then re-enable the Log Publisher for the changes to take effect.

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

```
  --set enabled:false
```

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

3. To validate a signed file, use the **`validate-file-signature`** tool to check if a signed file has been altered.

```
$ bin/validate-file-signature --file logs/audit
```

```
All signature information in file 'logs/audit' is valid
```

If any validation errors occur, a message displays that is similar to this:

```
One or more signature validation errors were encountered while validating
the contents of file 'logs/audit':
* The end of the input stream was encountered without encountering the end
of an active signature block. The contents of this signed block cannot be
trusted because the signature cannot be verified
```

Configure log retention and log rotation policies

PingData servers enable configuring log rotation and log retention policies.

Log retention

When any retention limit is reached, the server removes the oldest archived log prior to creating a new log. Log retention is only effective if a log rotation policy is in place.

A new log publisher must have at least one log retention policy configured. The following policies are available:

File Count Retention policy

Sets the number of log files for the server to retain. The default file count is 10 logs. If the file count is set to 1, the log will continue to grow indefinitely without being rotated.

Free Disk Space Retention policy

Sets the minimum amount of free disk space. The default free disk space is 500 MB.

Size Limit Retention policy

Sets the maximum size of the combined archived logs. The default size limit is 500 MB.

Custom Retention policy

Create a new retention policy that meets the server's requirements. This will require developing custom code to implement the desired log retention policy.

Never Delete Retention policy

Used in a rare event that does not require log deletion.

Log rotation

When a rotation limit is reached, the server rotates the current log and starts a new log. A new log publisher must have at least one log rotation policy configured. The following policies are available:

Time Limit Rotation policy

Rotates the log based on the length of time since the last rotation. Default implementations are provided for rotation every 24 hours and every seven days.

Fixed Time Rotation policy

Rotates the logs every day at a specified time (based on 24-hour). The default time is 2359.

Size Limit Rotation policy

Rotates the logs when the file reaches the maximum size. The default size limit is 100 MB.

Never Rotate policy

Used in a rare event that does not require log rotation.

Configure the log rotation policy

Use **dsconfig** to modify the log rotation policy for the access logger:

```
$ bin/dsconfig set-log-publisher-prop \
  --publisher-name "File-Based Access Logger" \
  --remove "rotation-policy:24 Hours Time Limit Rotation Policy" \
  --add "rotation-policy:7 Days Time Limit Rotation Policy"
```

Configure the log retention policy

Use **dsconfig** to modify the log retention policy for the access logger:

```
$ bin/dsconfig set-log-publisher-prop \
  --publisher-name "File-Based Access Logger" \
  --set "retention-policy:Free Disk Space Retention Policy"
```

Configure log listeners

The server provides two log file rotation listeners: the copy log file rotation listener and the summarize log file rotation listener, which can be enabled with a log publisher. Log file rotation listeners allow the server to perform a task on a log file as soon as it has been rotated out of service. Custom log file listeners can be created with the Server SDK.

The copy log file rotation listener can be used to compress and copy a recently-rotated log file to an alternate location for long-term storage. The original rotated log file will be subject to deletion by a log file retention policy, but the copy will not be automatically removed.

Use the following command to create a new copy log file rotation listener:

```
$ dsconfig create-log-file-rotation-listener \
  --listener-name "Copy on Rotate" \
  --type copy \
  --set enabled:true \
  --set copy-to-directory:/path/to/archive/directory \
  --set compress-on-copy:true</screen>
```

The path specified by the `copy-to-directory` property must exist, and the file system containing that directory must have enough space to hold all of the log files that will be written there. The server will automatically monitor free disk space on the target file system and will generate administrative alerts if the amount of free space gets too low.

The summarize log file rotation listener invokes the **summarize-access-log** tool on a recently rotated log file and writes its output to a file in a specified location.

This provides information about the number and types of operations processed by the server, processing rates and response times, and other useful metrics. Use this with access loggers that log in a format that is compatible with the **summarize-access-log** tool, including the `file-based-access` and

`operation-timing-access` logger types. Use the following command to create a new summarize log file rotation listener:

```
$ dsconfig create-log-file-rotation-listener \
  --listener-name "Summarize on Rotate" \
  --type summarize \
  --set enabled:true \
  --set output-directory:/path/to/summary/directory
```

The summary output files have the same name as the rotated log file, with an extension of `.summary`. If the `output-directory` property is specified, the summary files are written to that directory. If not specified, files are placed in the directory in which the log files are written.

As with the copy log file rotation listener, summary files are not automatically be deleted. Though files are generally small in comparison to the log files themselves, make sure that there is enough space available in the specified storage directory. The server automatically monitors free disk space on the file system to which the summary files are written.

System alarms, alerts, and gauges

An alarm represents a stateful condition of the server or a resource that may indicate a problem, such as low disk space or external server unavailability. A gauge defines a set of threshold values with a specified severity that, when crossed, cause the server to enter or exit an alarm state. Gauges are used for monitoring continuous values like CPU load or free disk space (Numeric Gauge), or an enumerated set of values such as 'server available' or ‘server unavailable’ (Indicator Gauge). Gauges generate alarms, when the gauge’s severity changes due to changes in the monitored value. Like alerts, alarms have severity (NORMAL, WARNING, MINOR, MAJOR, CRITICAL), name, and message. Alarms will always have a `Condition` property, and may have a Specific Problem or Resource property. If surfaced through SNMP, a `ProbableCause` property and `AlarmType` property are also listed. Alarms can be configured to generate alerts when the alarm's severity changes.

There are two alert types supported by the server - standard and alarm-specific. The server constantly monitors for conditions that may need attention by administrators, such as low disk space. For this condition, the standard alert is `low-disk-space-warning`, and the alarm-specific alert is `alarm-warning`. The server can be configured to generate alarm-specific alerts instead of, or in addition to, standard alerts. By default, standard alerts are generated for conditions internally monitored by the server. However, gauges can only generate alarm-alerts.

The server installs a set of gauges that are specific to the product and that can be cloned or configured through the **`dsconfig`** tool. Existing gauges can be tailored to fit each environment by adjusting the update interval and threshold values. Configuration of system gauges determines the criteria by which alarms are triggered. The Stats Logger can be used to view historical information about the value and severity of all system gauges.

PingData servers are compliant with the International Telecommunication Union CCITT Recommendation X.733 (1992) standard for generating and clearing alarms. If configured, entering or exiting an alarm state can result in one or more alerts. An alarm state is exited when the condition no longer applies. An `alarm_cleared` alert type is generated by the system when an alarm's severity changes from a non-normal severity to any other severity. An `alarm_cleared` alert will correlate to a previous alarm when Condition and Resource property are the same. The Alarm Manager, which governs the actions performed when an alarm state is entered, is configurable through the **`dsconfig`** tool and Administrative Console.

Like the Alerts Backend, which stores information in `cn=alerts`, the Alarm Backend stores information within the `cn=alarms` backend. Unlike alerts, alarm thresholds have a state over time that can change in severity and be cleared when a monitored value returns to normal. Alarms can be viewed with the **`status`** tool. As with other alert types, alert handlers can be configured to manage the alerts generated by alarms. A complete listing of system alerts, alarms, and their severity is available in `<server-root>/docs/admin-alerts-list.csv`.

Alert handlers

Alert notifications can be sent to administrators when significant problems or events occur during processing, such as problems during server startup or shutdown. The server provides a number of alert handler implementations configured with the **`confined`** tool, including:

Error Log Alert Handler

Sends administrative alerts to the configured server error logger(s).

JMX Alert Handler

Sends administrative alerts to clients using the Java Management Extensions (JMX) protocol. The server uses JMX for monitoring entries and requires that the JMX connection handler be enabled.

SNMP Alert Handler

Sends administrative alerts to clients using the Simple Network Monitoring Protocol (SNMP). The server must have an SNMP agent capable of communicating via SNMP 2c.

If needed, the Server SDK can be used to implement additional, third-party alert handlers.

Configure alert handlers

Alert handlers can be configured with the **`dsconfig`** tool. PingData servers support JMX, SMTP, and SNMP. Use the `--help` option for a list of configuration options. The following is a sample command to create and enable an SMTP Alert handler from the command line:

```
$ bin/dsconfig create-alert-handler \
  --handler-name "SMTP Alert Handler" \
  --type smtp \
  --set enabled:true \
  --set "sender-address:alerts@example.com" \
  --set "recipient-address:administrators@example.com" \
  --set "message-subject:Directory Admin Alert \%\%alert-type\%\%" \
  --set "message-body:Administrative alert:\\n\%\%alert-message\%\%"
```

Testing alerts and alarms

About this task

After alarms and alert handlers are configured, verify that the server takes the appropriate action when an alarm state changes by manually increasing the severity of a gauge. Alarms and alerts can be verified with the **`status`** tool.

Steps

1. Configure a gauge with **`dsconfig`** and set the `override-severity` property to critical. The following example uses the CPU Usage (Percent) gauge.

   ```
   $ dsconfig set-gauge-prop \
     --gauge-name "CPU Usage (Percent)" \
     --set override-severity:critical
   ```

2. Run the **`status`** tool to verify that an alarm was generated with corresponding alerts. The **`status`** tool provides a summary of the server's current state with key metrics and a list of recent alerts and alarms. The sample output has been shortened to show just the alarms and alerts information.

   ```
   $ bin/status
   ```

   ```
                            --- Administrative Alerts ---
   ```

```
Severity : Time           : Message
---------:---------------:-------------------------------------------------
Error    : 11/Aug/2016    : Alarm [CPU Usage (Percent). Gauge CPU Usage
(Percent)
         : 15:41:00 -0500 : for Host System has
         :                : a current value of '18.583333333333332'.
         :                : The severity is currently OVERRIDDEN in the
         :                : Gauge's configuration to 'CRITICAL'.
         :                : The actual severity is: The severity is
         :                : currently 'NORMAL', having assumed this
severity
         :                : Mon Aug 11 15:41:00 CDT 2016. If CPU use is
high,
         :                : check the server's current workload and make
any
         :                : needed adjustments. Reducing the load on the
system
         :                : will lead to better response times.
         :                : Resource='Host System']
         :                : raised with critical severity
Shown are alerts of severity [Info,Warning,Error,Fatal] from the past 48
hours
Use the --maxAlerts and/or --alertSeverity options to filter this list
```

```
--- Alarms ---
Severity : Severity   : Condition : Resource    : Details
         : Start Time :           :              :
---------:------------:-----------:--------------:-------------------------
Critical : 11/Aug/2016: CPU Usage : Host System : Gauge CPU Usage
(Percent) for
         : 15:41:00   : (Percent) : : Host System
         :  -0500     :           :              : has a current value of
         :            :           :              : '18.785714285714285'.
         :            :           :              : The severity is
currently
         :            :           :              : 'CRITICAL', having
assumed
         :            :           :              : this severity Mon Aug 11
         :            :           :              : 15:49:00 CDT 2016. If
CPU use
         :            :           :              : is high, check the
server's
         :            :           :              : current workload and
make any
         :            :           :              : needed adjustments.
Reducing
         :            :           :              : the load on the system
will
         :            :           :              : lead to better response
times
Shown are alarms of severity [Warning,Minor,Major,Critical
Use the --alarmSeverity option to filter this list
```

Use the status tool

PingData servers provides the **`status`** tool, which outputs the health of the server. The **`status`** tool polls the current health of the server and displays summary information about the number of operations processed in the network. The tool provides the following information:

Status tool sections

Status section	Description
Server Status	Displays the server start time, operation status, number of connections (open, max, and total).
Server Details	Displays the server details including host name, administrative users, install path, server version, and Java version.
Connection Handlers	Displays the state of the connection handlers including address, port, protocol and current state.
Admin Alerts	Displays the 15 administrative alerts that were generated over the last 48-hour period. Limit the number of displayed alerts using the `--maxAlerts` option. For example, **`status --maxAlerts 0`** suppresses all alerts.

Synchronization-specific status

The **`status`** tool displays the following information for PingDataSync Server.

PingDataSync Server status information

Status section	Description
Sync Topology	Displays information about the connected Sync topology and any standby PingDataSync Server instances.
Summary for Sync Pipe	Displays the health status for each Sync Pipe configured on the topology. Status for each Sync Pipe includes the following: ▪ Started – Indicates whether the Sync Pipe has started. ▪ Current Ops Per Second – Lists the current throughput rate in operations per second. ▪ Percent Busy – Lists the number of current operations currently divided by the number of worker threads. ▪ Changes Detected – Lists the total number of changes detected. ▪ Ops Completed Total – Lists the total number of changes detected and completed. ▪ Num Ops In Flight – Lists the number of operations that are in flight. ▪ Num Ops In Queue – Lists the number of operations that are on the input queue waiting to be synchronized. ▪ Source Unretrieved Changes – Lists how many outstanding changes are still in the source change log that have not yet been retrieved by PingDataSync Server. If this is greater than zero, it indicates a backlog, because the internal queue is too full to include these changes. ▪ Failed Op Attempts – Lists the number of failed operation attempts. ▪ Poll For Source Changes Count – Lists the number of times that the source has been polled for changes.

Status section	Description
Operations completed for the Sync Pipe	Displays the completed operation statistics for the sync pipe, including the following: ▪ Success – Lists the number of changes that completed successfully. ▪ Out Of Scope – Lists the number of changes that were included in the Sync Pipe, but were dropped because they did not match criteria in a Sync Class. ▪ Op Type Not Synced – Lists the number of changes that completed because the operation type is not synchronized. ▪ No Change Needed – Lists the number of changes that completed because no change was needed. ▪ Entry Already Exists – Lists the number of changes that completed unsuccessfully because the entry already existed for a Create operation. ▪ No Match Found – Lists the number of changes that completed unsuccessfully because no match for an operation (such as Modify) was found. ▪ Multiple Matches Found – Lists the number of changes that completed unsuccessfully because multiple matches for a source entry were found at the destination. ▪ Failed During Mapping – Lists the number of changes that completed unsuccessfully because there was a failure during attribute or DN mapping. ▪ Failed At Resource – Lists the number of changes that completed unsuccessfully because they failed at the source. ▪ Unexpected Exception – Lists the number of changes that completed unsuccessfully because there was an unexpected exception during processing. ▪ Total – Lists the number of operations completed.
Sync Pipe source stats	Displays the source statistics for the external server, including the following: ▪ Is Connected – Indicates whether the Sync Source is connected or not. ▪ Connected Server – Indicates the host name and port number of the connected server. ▪ Successful Connect Attempts – Indicates the number of successful connection attempts. ▪ Failed Connect Attempts – Indicates the number of failed connection attempts. ▪ Forced Disconnects – Indicates the number of forced disconnects. ▪ Root DSE Polls – Indicates the number of polling attempts of the root DSE. ▪ Unretrieved Changes – Indicates the number of unretrieved changes. ▪ Entries Fetched – Indicates the number of entries fetched from the source. ▪ Failed To Decode Changelog Entry – Indicates the operations that failed to decode changelog entries. ▪ Ops Excluded By Modifiers Name – Indicates the number of operations excluded by modifier’s name. ▪ Num Backtrack Batches Retrieved – Indicates the number of backtrack batches retrieved.

Status section	Description
Sync Pipe destination stats	Displays the destination statistics for the external server, including the following: ▪ Is Connected – Indicates whether the Sync Source is connected or not. ▪ Connected Server – Indicates the connection URL of the connected server. ▪ Successful Connect Attempts – Indicates the number of successful connection attempts. ▪ Failed Connect Attempts – Indicates the number of failed connection attempts. ▪ Forced Disconnects – Indicates the number of forced disconnects. ▪ Entries Fetched – Indicates the number of entries fetched. ▪ Entries Created – Indicates the number of entries created. ▪ Entries Modified – Indicates the number of entries modified. ▪ Entries Deleted – Indicates the number of entries deleted.

StatsD monitoring endpoint

The Monitoring Endpoint configuration type provides the StatsD Endpoint type that you can use to transfer metrics data in the StatsD format.

Examples of metrics you can send are:

- Busy worker thread count
- Garbage collection statistics
- Host system metrics such as CPU and memory

For a list of available metrics, use the interactive **`dsconfig`** menu for the Stats Collector Plugin or in the Administrative Console, edit the **Stats Collector** plugin as explained in the second example.

You configure the Monitoring Endpoint using the **`dsconfig`** command. When you configure Monitoring Endpoint, you include:

- The endpoint's hostname
- The endpoint's port
- A toggle to use TCP or UDP
- A toggle to use SSL if you use TCP

For example, to configure a new StatsD Monitoring Endpoint to send UDP data to localhost port 8125 using **`dsconfig`**:

```
dsconfig create-monitoring-endpoint \
    --type statsd \
    --endpoint-name StatsDEndpoint \
    --set enabled:true \
    --set hostname:localhost \
    --set server-port:8125 \
    --set connection-type:unencrypted-udp
```

If you are using the Administrative Console:

1. Click **Show Advanced Configuration**.
2. In the **Logging, Monitoring, and Notifications** section, click **Monitoring Endpoints**.
3. Click **New Monitoring Endpoint**.

You can send data to any number of monitoring endpoints.

The Stats Collector Plugin controls the metrics used by the StatsD monitoring endpoint. To send metrics with the StatsD monitoring endpoint, you must enable the Stats Collector Plugin. Also, you must configure the Stats Collector Plugin to indicate the metrics to send.

To enable the Stats Collector Plugin or to configure the type of data sent, use the **dsconfig** command or the Administrative Console. For example, to enable the Stats Collector Plugin to send host CPU metric, memory metrics, and server status metrics using **dsconfig**:

```
dsconfig set-plugin-prop \
    --plugin-name "Stats Collector" \
    --set enabled:true \
    --set host-info:cpu \
    --set host-info:disk \
    --set status-summary-info:basic
```

If you are not using Data Metrics Server to monitor your server, you can disable the generation of some metrics files that are not necessary for the StatsD Monitoring Endpoint. To do this, set the `generate-collector-files` property on the Stats Collector Plugin to `false`.

If you are using the Administrative Console:

1. Click **Show Advanced Configuration**.
2. In the **LDAP (Administration and Monitoring)** section, click **Plugin Root**
3. Edit the **Stats Collector** plugin.

After you enable the Stats Collector and create the StatsD monitoring endpoint, you can:

- Use the data with Splunk as explained in *Sending Metrics to Splunk with StatsD* on page 758.
- Configure other tools that support StatsD, such as CloudWatch or a Prometheus StatsD exporter, to use the data. For more information about this configuration, see your tool's StatsD documentation. Configure the StatsD monitoring endpoint to use the correct host and port. The `dsconfig create-monitoring-endpoint` example above uses a host of localhost and a port of 8125. You can also set these values in the Administrative Console.

Sending Metrics to Splunk with StatsD

About this task

With the StatsD Endpoint type, you can send metric data to a Splunk installation.

In Splunk, you can use SSL to secure ports that are open for StatsD.

> ⓘ **Note:**
>
> StatsD metrics are typically sent over UDP. By using UDP, the client sending metrics does not have to block as it would if using TCP. However, using TCP guarantees order and ensures no metrics are lost.

You can configure open UDP (or TCP) ports in Splunk to accept only connections from a certain hostname or IP address.

To securely send UDP (or TCP) data to Splunk, you can:

Steps

1. Send the data to a Splunk Universal Forwarder.
2. Have the forwarder communicate with the Splunk Indexer over SSL.

Monitor PingDataSync Server

PingDataSync Server exposes its monitoring information under the `cn=monitor` entry. Various tools can be used to surface the server's information including the PingDataMetrics Server, the Administrative Console, JConsole, LDAP command-line tools, or SNMP. The following information is collected for PingDataSync Server. To configure the PingData PingDataMetrics Server to display PingDataSync Server data, see the Ping Identity PingDataMetrics Server Administration Guide.

PingDataSync Server monitoring component

Component	Description
Active Operations	Provides information about the operations currently being processed by the server including the number of operations, information about the operation, and the number of active persistent searches.
Backend	Provides general information about the state of the server backend, including the backend ID, base DN(s), entry counts, entry count for the `cn=admin` data, writability mode, and whether it is a private backend. The following backend monitors are provided: ▪ adminRoot ▪ ads-truststore ▪ alerts ▪ backup ▪ config ▪ monitor ▪ schema ▪ tasks ▪ userRoot
Berkeley DB JE Environment	Provides information about the state of the Oracle Berkeley DB Java Edition database used by the PingDataSync Server backend.
Client Connections	Provides information about all client connections to the server.
Disk Space Usage	Provides information about the disk space available to various components of the server. The disk space usage monitor evaluates the free space at locations registered through the `DiskSpaceConsumer` interface. Disk space monitoring excludes disk locations that do not have server components registered. However, other disk locations may still impact server performance, such as the operating system disk, if it becomes full. When relevant to the server, these locations include the server root, the location of the `config` directory, the location of every log file, all JE backend directories, the location of the changelog, the location of the replication environment database, and the location of any server extension that registers itself with the `DiskSpaceConsumer` interface.
Connection Handler	Provides information about the available connection handlers on the server, which include the LDAP and LDIF connection handlers. These handlers are used to accept client connections.
General	Provides general information about the server, including product name and server version.
JVM Stack Trace	Provides a stack trace of all threads processing within the JVM.
LDAP Connection Handler Statistics	Provides statistics about the interaction that the associated LDAP connection handler has had with its clients, including the number of connections established and closed, bytes read and written, LDAP messages, and operations handled.
Processing Time Histogram	Categorizes operation processing times into a number of user-defined buckets of information, including the total number of operations processed, overall average response time, and number of processing times between `0ms` and `1ms`.

Component	Description
System Information	Provides general information about the system and the JVM on which the server is running, including host name, operating system, JVM architecture, Java home, and Java version.
Version	Provides information about the product version, including build ID and revision number.
Work Queue	Provides information about the state of the PingDataSync Server work queue, which holds requests until they can be processed by a worker thread. The work queue configuration has a `monitor-queue-time` property set to `true` by default. This logs messages for new operations with a `qtime`attribute included in the log messages. Its value is expressed in milliseconds and represents the length of time that operations are held in the work queue. A dedicated thread pool can be used for processing administrative operations. This thread pool enables diagnosis and corrective action if all other worker threads are processing operations. To request that operations be processed using the administrative thread pool, the requester must have the `use-admin-session`privilege (included for root users). By default, eight threads are available for this purpose. This can be changed with the `num-administrative-session-worker-threads` property in the work queue configuration.

DevOps and Infrastructure as Code

Derived from the words *development* and *operations*, the term *DevOps* refers to the practices that a company follows to ensure the production of high-quality products, while also minimizing the amount of time between the commitment of a system change and the implementation of that change in a production environment.

Most companies practice one of the following service models:

- Pets — With the *pets service model*, servers are built and managed manually, and are treated as unique and indispensable. Examples include mainframes, database systems, and load balancers.
- Cattle — With the *cattle service model*, arrays of multiple replaceable servers are built with automated tools. During a failure event, an array automatically restarts failed services and replicates data. Examples include web server arrays, search clusters, and multi-master datastores.

Historically, servers have been treated like pets. The failure of one or multiple servers was often viewed as an emergency, and extensive resources were usually required to repair the damage. In the new DevOps paradigm, servers are recognized as dispensable and treated like cattle. When a server fails, the infrastructure replaces it immediately, configuring the replacement server identically to the failed one. Because no human intervention is required to fix them, the servers are considered self-healing.

To help treat your servers more like cattle than pets, PingDataSync Server supports server profiles and features like topology-management tools. For example, the **`manage-profile setup`** represents a single command that performs all the steps from the pet service model on a `server-profile` directory, a well-defined directory structure with all the necessary server configuration bits. Similarly, the **`manage-profile replace-profile`** command performs similar steps with a single command invocation after a server is updated to a new version.

The scripts in the `server-profile` directory are declarative of the environment. Consequently, what you define in the `server-profile` directory is what you get on the servers. No one needs to identify a server's current configuration and compute the differences that must be applied to attain the appropriate end state. Prior to server profiles, this problem was difficult to address, especially where no history was available. In such scenarios, an administrator might have needed to obtain the current configuration from

the servers, to manually compute the difference between the current and desired configuration, and to apply the configuration changes, hoping all the while that nobody had changed the configuration during the process. Server profiles eliminate this procedural approach to applying configuration changes, and they simplify the steps associated with determining what is deployed in an environment. For more information on server profiles, see *Server profiles* on page 816.

Another principle that relates closely to DevOps is infrastructure as code (IaC), the concept of managing your operations in the same manner as your application and other code for general release with proper versioning, continuous integration, quality control, and release cycles. Customers who deploy PingDataSync Servers as pets today can take advantage of current DevOps and IaC principles to turn them into cattle.

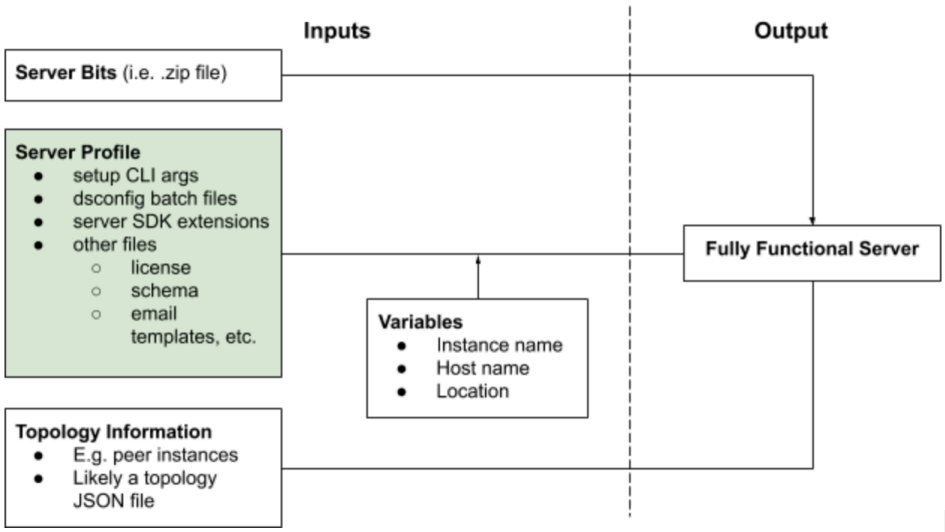
Server profiles

Regardless of the service model that your company follows, *server profiles* help you achieve your goals. At a basic level, a server profile defines a format for the configuration of a server by combining the following files into a single concrete structure:

- **`dsconfig`**
- Setup arguments
- Server SDK extensions
- Additional miscellaneous files

The primary goal of a server profile is to simplify the deployment of PingDataSync Server and related products by using deployment automation frameworks. When products support this capability, the amount of scripting that is required across automation frameworks — like Docker, Kubernetes, and Ansible — is reduced considerably.

The following image shows the role that a server profile plays in building a fully functional running server.



As a declarative form of a full server configuration, a server profile provides the following advantages:

- Provides a more complete and easily comparable way to define an individual server's configuration. Changes between different servers are easier to understand, and incremental changes to a server's configuration are easier to track.
- Ensures that each server instance is configured identically to its peers.
- Can be applied directly to new as well as previously installed instances.
- Shares a common configuration across a deployment pipeline of development, test, and production environments without unnecessary duplication. For information about substituting variables that differ by environment, see *Variable substitution* on page 817.
- Facilitates deployment automation by representing configuration as code.

- Reduces the number of additional configuration steps that are required to place a server into production.
- Makes the execution of various configuration changes more consistent and repeatable. The strategy of using a server profile to represent the final state of a server is less error-prone than recording a step-by-step process to attain that state.
- Can be managed easily in a version-control system.
- Simplifies the management of servers outside deployment automation frameworks.

A continuous deployment workflow can work with server profiles to make certain that changes to a server profile are moved automatically into production. In a stateful environment, the **`manage-profile replace-profile`** subcommand can be used to update existing servers. In a stateless environment, in which servers are considered immutable, **`manage-profile setup`** can be used to deploy new servers whenever a profile changes. With multiple environments, this deployment can be performed in a test environment before moving to production.

When working with zipped server SDK extensions and other files that might not be stored in a version-control system, the server profile can be modified to include these files prior to its use. For example, if the code for an extension is stored in a separate repository, it can be built and dropped into the server profile immediately before the **`manage-profile`** tool is run. This process is part of the deployment automation logic that uses the server profile, and it can be followed for any files that are needed by the server profile but whose versions are not controlled.

For more information about the **`manage-profile`** tool, see *About the manage-profile tool* on page 820.

Variable substitution

You can use the **`manage-profile`** tool to substitute different variables in server profiles.

The **`manage-profile`** tool uses the format **`${VARIABLE}`** to support the substitution of variables in profiles. This format can be escaped by using another **`$`**. For example, after substitution, **`$${VARIABLE}`** becomes **`${VARIABLE}`**.

Variable values can be read from a profile variables file or from environment variable values. If both options are used, the values that are specified in the file overwrite any environment variables.

The following code provides an example of how you can set user-defined variables by using a variables file in the server profile.

```
HOSTNAME=testserver.example.com
PORT=389
```

The following table describes built-in variables that can also be referenced in the server profile. Use these variables in the format previously described.

Built-in variable	Description
`PING_SERVER_ROOT`	Evaluates to the absolute path of the server's root directory
`PING_PROFILE_ROOT`	Evaluates to the individual profile's root directory ⓘ **Note:** Use PING_PROFILE_ROOT only with files that are not needed after initial setup, such as password files in `setup-arguments.txt`. Do not use the PING_PROFILE_ROOT variable for files needed while the server is running. The **`manage-profile`** tool creates a temporary copy of the server profile that is deleted after the tool completes, so files are not accessible under PING_PROFILE_ROOT

Built-in variable	Description
	when the server is running. For files you need while the server is running, such as keystore and truststore files, copy the files into the server root using the profile's `server-root/pre-setup` directory, and then refer to the files using with the PING_SERVER_ROOT variable.

For more information about the tool's usage, run the command **`bin/manage-profile --help`**.

Profile Structure

Use either of the following methods to create a server profile:

- Use the template named `server-profile-template.zip`, which is located in the `resource/` directory.
- Run the **`manage-profile generate-profile`** subcommand. The **`manage-profile`** tool references a file system directory structure rather than a ZIP file.

Files can be added to each directory as needed.

The following hierarchy represents the file structure of a basic server profile:

```
-server-profile/
  |-- dsconfig/
  |-- misc-files/
  |-- server-root/
  |   |-- post-setup/
  |   |-- pre-setup/
  |-- server-sdk-extensions/
  |-- setup-arguments.txt
  |-- variables-ignore.txt
```

setup-arguments.txt

When creating a profile, the first step is to add arguments to the file `setup-arguments.txt`. When **`manage-profile`** setup is run, these arguments are passed to the server's setup tool. To view the arguments that are available in this file, run the server's **`setup --help`** command.

To provide the equivalent, non-interactive CLI arguments after any prompts have been completed, run **`setup`** interactively. The `setup-arguments.txt` file in the profile template contains an example set of arguments that can be changed.

`setup-arguments.txt` is the only required file in the profile.

dsconfig/

`dsconfig` batch files can be added to the `dsconfig` directory. These files, each of which must include a `.dsconfig` extension, contain **`dsconfig`** commands to apply to server.

Because the **`dsconfig`** batch files are ordered lexicographically, `00-base.dsconfig` runs `before 01-second.dsconfig`, and so on.

To produce a `dsconfig` batch file that reproduces the current configuration, run **`bin/config-diff`**.

server-root/

Any server root files can be added to the `server-root` directory, including schema files, email template files, custom password dictionaries, and other files that must be present on the final server root. Add these files to the `server-root/pre-setup` or `server-root/post-setup` directory, depending on when

they need to be copied to the server root. Most server root files are added to the server-root/pre-setup directory.

server-sdk-extensions/

Add server SDK extension `.zip` files to the `server-sdk-extensions` directory. Include any configuration that is necessary for the extensions in the profile's **`dsconfig`** batch files.

variables-ignore.txt

The `variables-ignore.txt` file is an optional component of the server profile. It is useful when adding bash scripts to the server root because such files often contain expressions that the **`manage-profile`** tool normally interprets as variables.

Add `variables-ignore.txt` to a profile's root directory to indicate the relative paths of any files that are not to have their variables substituted.

The following example shows the contents of a typical `variables-ignore.txt` file:

```
server-root/pre-setup/script-to-ignore.sh
server-root/post-setup/another-file-to-ignore.txt
```

server-root/permissions.properties

The `permissions.properties` file, located in the `server-root` directory, is an optional file that specifies the permissions to apply to files that are copied to the server root. These permissions are represented in octal notation. By default, server root files maintain their permissions when copied.

The following example shows the contents of a typical `permissions.properties` file:

```
default=700
file-with-special-permissions.txt=600
new-subdirectory/file-with-special-permissions.txt=644
bin/example-script.sh=760
```

misc-files/

Additional documentation and other files can be added to the `misc-files` directory, which the **`manage-profile`** tool does not use. Use the variable **`PING_PROFILE_ROOT`** to refer to files in this directory from other locations, such as `setup-arguments.txt`.

> ⓘ **Note:**
>
> Use PING_PROFILE_ROOT only with files that are not needed after initial setup, such as password files in `setup-arguments.txt`. Do not use the PING_PROFILE_ROOT variable for files needed while the server is running. The **`manage-profile`** tool creates a temporary copy of the server profile that is deleted after the tool completes, so files are not accessible under PING_PROFILE_ROOT when the server is running. For files you need while the server is running, such as keystore and truststore files, copy the files into the server root using the profile's `server-root/pre-setup` directory, and then refer to the files using with the PING_SERVER_ROOT variable.

For example, a password file named `password.txt` in the `misc-files` directory could be referenced with `${PING_PROFILE_ROOT}/misc-files/password.txt` in `setup-arguments.txt`. Use a reference like this example to supply the file for the **`--rootUserPasswordFile`** argument in `setup-arguments.txt`.

About the manage-profile tool

The **manage-profile** tool is provided with the server to work with server profiles. It includes subcommands for creating, applying, and replacing server profiles, all of which significantly reduce the effort required by an administrator to configure a server appropriately.

The following sections describe these subcommands in more detail. For more information about the **manage-profile** tool, run **manage-profile --help**. For more information about each individual subcommand and its options, run **manage-profile <subcommand> --help**.

manage-profile generate-profile

To create a server profile from a configured server, use the **generate-profile** subcommand. The generated profile contains the following information, which provides a base for completing a profile:

- Command-line arguments that were used to set up the server
- **dsconfig** commands necessary to configure the server
- Installed server SDK extensions
- Files that are added to the server root

To produce a complete profile, some parts of the generated profile might require modifications, such as adding password files that `setup-arguments.txt` uses. The **--instanceName** and **--localHostName** arguments in `setup-arguments.txt` are made variables by **generate-profile**, and must be provided values when other **manage-profile** subcommands use the generated profile.

manage-profile setup

To apply a server profile to a fresh, unconfigured server, use the **setup** subcommand, which replaces the normal setup tool when using a server profile. Run **manage-profile setup** to complete the following tasks:

- Copies the pre-setup files to the server root
- Runs the setup tool
- Copies the post-setup files to the server root
- Installs any server SDK extensions
- Runs any dsconfig batch files
- Imports any LDIF files
- Installs the server SDK extensions
- Starts the server

While manage-profile setup is running, a copy of the profile is created in a temporary directory that can be specified by using the **--tempProfileDirectory** argument. The command leaves the server in a complete and running state when finished, unless the **--doNotStart** argument is specified.

manage-profile replace-profile

Run the **replace-profile** subcommand on a server that was originally set up with a server profile to replace its configuration with a new profile. The tool applies a specified server profile to an existing server while preserving its data, topology configuration, and replication configuration.

While **manage-profile replace-profile** is running, the existing server is stopped and moved to a temporary directory that the **--tempServerDirectory** argument can specify. A fresh, new server is subsequently installed and set up with the new profile. The final server is left running if it was running before the command was started, and remains stopped if it was stopped.

Run **manage-profile replace-profile** from a second unzipped server install package on the same host as the existing server, similar to the **update** tool. Use the **--serverRoot** argument to specify the root of the existing server that will have its profile replaced.

If files have been added or modified in the server root since the most recent **manage-profile setup** or **manage-profile replace-profile** was run, they are included in the final server with the replaced profile. Otherwise, files specifically added from the `server-root` directory of the previous server profile

are absent from the final server with the replaced profile. If errors occur during the subcommand, such as the new profile having an invalid `setup-arguments.txt` file, the existing server returns to its original state from before **`manage-profile replace-profile`** was run.

ⓘ **Note:** The **`manage-profile replace-profile`** tool can update the server version when needed.

ⓘ **Note:** The **`manage-profile replace-profile`** tool can directly apply configuration changes when there are no other changes in the new profile. This is a shorter process when making small changes to **`dsconfig`**.

Server Profiles in a Pets Service Model

Server profiles and other DevOps concepts are also invaluable in a pets service model. For example, the step of using the **`manage-profile generate-profile`** subcommand to generate a server profile from a production server creates an easily consumable representation of the server's configuration. In nearly every scenario, the generation of a profile from an existing server is simpler than the piecing together manually of schemas, extensions, and other configuration information to create an image of that server. Additionally, generated profiles can be backed up or checked in to source control to maintain a consistent picture of an active server's configuration.

Another valuable use of server profiles involves setting up servers in a test environment that is separate from production. For example, a profile that matches the profile of a production server can be generated and used to install a fresh test server that matches the production server. Further, variable substitution allows environmental changes, such as local host name or instance name, without requiring a separate profile. Because the server's original configuration matches the running production server, adjustments can be tested easily. This approach provides more consistency when you validate changes before moving them to production.

If a new pet server has been set up with a server profile, **`manage-profile replace-profile`** can be used to apply changes to the profile. Rather than using scripts or a manual process to apply individual changes, **`replace-profile`** provides a consistent, repeatable method of moving to a new server profile. This strategy automates more easily and is less prone to human error.

For more information about the **`manage-profile`** tool, see *About the manage-profile tool* on page 820.

Troubleshooting

There are several ways to troubleshoot issues with PingDataSync Server.

Synchronization troubleshooting

The majority of synchronization problems involve the connection state of the external servers and the synchronization of the data between the two endpoints. Make sure PingDataSync Server can properly fail over to another endpoint or server instance if the connection fails on the highest priority external server.

Another factor in troubleshooting synchronization is determining if the DN and attribute mappings were properly configured and if the information is properly being synchronized across the network. Typical scenarios include checking for any entry failures and mapping issues.

ⓘ **Note:**

Use the **`resync`** tool to validate Sync Classes and data mappings from one endpoint to another. The tool provides a `dry-run` option that verifies data operations without actually affecting the data.

The following log files are specific to PingDataSync Server, and contain details about the synchronization processes:

Sync Log

Provides information about the synchronization operations that occur within the server. Specifically, the Sync Log records all changes applied, detected or failed; dropped operations that were not synchronized; changes dropped due to being out of scope, or no changes needed for synchronization. The log also shows the entries that were involved in the synchronization process.

Sync Failed Operations Log

Provides a list of synchronization operations that have failed.

Resync Log

Provides summaries or details of synchronized entries and any missing entries in the Sync Destination.

Resync Error Log

Provides error information for **`resync`** operations.

Management tools

Each PingData server provides command-line tools to manage, monitor, and diagnose server operations. Each tool provides a description of the subcommands, arguments, and usage examples needed to run the tool.

> ⓘ **Note:**
>
> For detailed information and examples of the command-line tools, see the Configuration Reference Guide in the `<server-root>/docs` directory, or linked from the Administrative Console.

To view detailed argument options and examples, use `--help` with the each tool:

```
$ bin/dsconfig --help
```

For those utilities that support additional subcommands (such as **`dsconfig`**), list the subcommands with the following:

```
$ bin/dsconfig --help-subcommands
```

View more detailed subcommand information by using `--help` with the specific subcommand:

```
$ bin/dsconfig list-log-publishers --help
```

Troubleshooting tools

PingData provides utilities to troubleshoot the state of each server and to determine the cause of any issues. The following tools are available for diagnosing any problems and are located in the `<server-root>/bin` directory, or the `<server-root>/bat` directory on Windows systems:

Troubleshooting tools

Tool	Description
status	Provides a high-level view of the current operational state of the server and displays any recent alerts that have occurred in past 24 hours.
ldap-diff	Used to compare one or more entries across two server endpoints to determine data issues.

Tool	Description
ldapsearch	Retrieves the full entries from two different servers to determine the exact content of an entry from each server.
logs	The logs directory provides important logs that should be used to troubleshoot or monitor any issue with the server. Logs include server-specific operations and the following general logs: ▪ Error Log – Provides information about warnings, errors, or significant events that occur within the server. ▪ Debug Log – Provides detailed information, if enabled, about processing performed by the server, including any exceptions caught during processing, detailed information about data read from or written to clients, and accesses to the underlying database. ▪ Access loggers – Provide information about LDAP operations processed within the server. This log only applies to operations performed in the server. This includes configuration changes, searches of monitor data, and bind operations for authenticating administrators using the command-line tools and the Administrative Console.
collect-support-data	Used to aggregate the results of various support tools data for the Ping Identity Support team to diagnose. For more information, see *Use the collect-support-data* tool.
config-diff	Generate a summary of the configuration changes in a local or remote server instance. The tool can be used to compare configuration settings when troubleshooting issues, or when verifying configuration settings on new servers.

Use the status tool

PingData servers provides the **`status`** tool,which outputs the health of the server. The **`status`** tool polls the current health of the server and displays summary information about the number of operations processed in the network. The tool provides the following information:

Status tool sections

Status section	Description
Server Status	Displays the server start time, operation status, number of connections (open, max, and total).
Server Details	Displays the server details including host name, administrative users, install path, server version, and Java version.
Connection Handlers	Displays the state of the connection handlers including address, port, protocol and current state.
Admin Alerts	Displays the 15 administrative alerts that were generated over the last 48-hour period. Limit the number of displayed alerts using the `--maxAlerts` option. For example, status `-- maxAlerts 0` suppresses all alerts.

Using the collect-support-data tool

About this task

PingData servers provide information about their current state and any problems encountered. If a problem occurs, run the **`collect-support-data`** tool in the `/bin` directory. The tool aggregates all relevant

support files into a zip file that can be sent to a support provider for analysis. The tool also runs data collector utilities, such as **`jps`**, **`jstack`**, and **`jstat`** plus other diagnostic tools for the operating system.

The tool may only archive portions of certain log files to conserve space, so that the resulting support archive does not exceed the typical size limits associated with e-mail attachments.

The data collected by the **`collect-support-data`** tool may vary between systems. The data collected includes the configuration directory, summaries and snippets from the `logs` directory, an LDIF of the monitor and RootDSE entries, and a list of all files in the server root.

Steps

1. Navigate to the server root directory.
2. Run the **`collect-support-data`** tool. Include the host, port number, bind DN, and bind password.

```
$ bin/collect-support-data \
--hostname 100.0.0.1 --port 389 \
--bindDN "cn=Directory Manager"
--bindPassword secret \
--serverRoot /opt/PingData<server> \
--pid 1234
```

3. Email the zip file to a support provider.

Use the Sync log

The Sync log, located in the logs directory (`<server-root>/logs/sync`), provides useful troubleshooting information on the type of operation that was processed or completed. Most log entries provide the following common elements in their messages:

Sync log elements

Sync log element	Description
category	Indicates the type of operation, which will always by SYNC.
severity	Indicates the severity type of the message: INFORMATION, MILD_WARNING, SEVERE_WARNING, MILD_ERROR, SEVERE_ ERROR, FATAL_ERROR, DEBUG, or NOTICE.
msgID	Specifies the unique ID number assigned to the message.
op	Specifies the operation number specific to PingDataSync Server.
changeNumber	Specifies the change number from the source server assigned to the modification.
replicationCSN	Specifies the replication change sequence number from the source server.
replicaID	Specifies the replica ID from the source server if there are multiple backend databases.
pipe	Specifies the Sync Pipe that was used for this operation.
msg	Displays the result of this operation.

Sync log example 1

The following example displays an informational message that a modification to an entry was detected on the source server.

```
$ tail -f logs/sync
[17/May/2015:15:46:19 -0500] category=SYNC severity=INFORMATION
msgID=1893728293
```

```
op=14 changeNumber=15 replicationCSN=00000128A7E3C7D31E960000000F
replicaID=7830
pipe="DS1 to DS2" msg="Detected MODIFY of uid=user.993,ou=People,dc=example,
dc=com at ldap://server1.example.com:1389"
```

Sync log example 2

The next example shows a successful synchronization operation that resulted from a MODIFY operation on the source server and synchronized to the destination server.

```
[18/May/2015:13:54:04 -0500] category=SYNC severity=INFORMATION
msgID=1893728306 op=701
changeNumber=514663 replicationCSN=00000128ACC249A31E960007DA67
 replicaID=7830
pipe="DS1 to DS2" class="DEFAULT" msg="Synchronized MODIFY of
uid=user.698,ou=People,
dc=example,dc=com at ldap://server1.example.com:1389 by modifying entry
uid=user.698,
ou=People,dc=example,dc=com at ldap://server3.example.com:3389"
```

Sync log example 3

The next example shows a failed synchronization operation on a MODIFY operation from the source server that could not be synchronized on the destination server. The log displays the LDIF-formatted modification that failed, which came from a schema violation that resulted from an incorrect attribute mapping (`telephoneNumber -> telephone`) from the source to destination server.

```
[18/May/2015:11:29:49 -0500] category=SYNC severity=SEVERE_WARNING
msgID=1893859389
op=71831 changeNumber=485590 replicationCSN=00000128AC3DE8D51E96000768D6
replicaID=7830 pipe="DS1 to DS2" class="DEFAULT" msg="Detected MODIFY of
uid=user.941,ou=People,dc=example,dc=com at ldap://server1.example.com:1389,
but
failed to apply this change because: Failed to modify entry uid=user.941,
ou=People,dc=example,dc=com on destination 'server3.example.com:3389'.
Cause: LDAPException(resultCode=65(object class violation), errorMessage='
Entry uid=user.941,ou=People,dc=example,dc=com cannot be modified because
 the
resulting entry would have violated the server schema: Entry
uid=user.941,ou=People,
dc=example,dc=com violates the Directory Server schema configuration because
it
includes attribute telephone which is not allowed by any of theobjectclasses
defined in that entry') (id=1893859386
ResourceOperationFailedException.java:125
Build revision=6226). Details: Source change detail:
dn: uid=user.941,ou=People,dc=example,dc=com
changetype: modify
replace: telephoneNumber
telephoneNumber: 027167170433915
-
replace: modifiersName
modifiersName: cn=Directory Manager,cn=Root DNs,cn=config
-
replace: modifyTimestamp
modifyTimestamp: 20131010020345.546Z
Equivalent destination changes:
dn: uid=user.941,ou=People,dc=example,dc=com
changetype: modify
replace: telephone
telephone: 818002279103216
Full source entry:
```

```
dn: uid=user.941,ou=People,dc=example,dc=com
objectClass: person
... (more output)
Mapped destination entry:
dn: uid=user.941,ou=People,dc=example,dc=com
telephone: 818002279103216
objectClass: person
objectClass: inetOrgPerson
... (more output) ...
```

Troubleshooting synchronization failures

While many PingDataSync Server issues are deployment-related and are directly affected by the hardware, software, and network structure used in the synchronization topology, most failures usually fall into one of the following categories:

Entry Already Exists

When an add operation was attempted on the destination server, an entry with the same DN already exists.

No Match Found

A match was not found at the destination based on the current Sync Classes and correlation rules (DN and attribute mapping). When this value has a high count, correlation rule problems are likely.

Failure at Resource

Indicates that some other error happened during the synchronization process that does not fall into the first two categories. Typically, these errors are communication problems with a source or destination server.

Statistics for these and other types of errors are kept in the `cn=monitor` branch and can be viewed directly using the **`status`** command.

Troubleshooting "Entry Already Exists" failures

About this task

If there is a count for the `EntryAlready Exists` statistic using the **`status`** tool, verify the problem in the sync log. For example, the **`status`** tool displays the following information:

```
           --- Ops Completed for 'DS1 to DS2' Sync Pipe ---
Op Result               : Count
-----------------------:------
Success                 : 0
Out Of Scope            : 0
Op Type Not Synced      : 0
No Change Needed        : 0
Entry Already Exists    : 1
No Match Found          : 1
Multiple Matches Found  : 0
Failed During Mapping   : 0
Failed At Resource      : 0
Unexpected Exception    : 0
Total                   : 2
```

Verify the change by viewing the `<server-root>/logs/sync` file to see the specific operation, which could be due to someone manually adding the entry on the target server:

```
op=2 changeNumber=529277 replicationCSN=00000128AD0D9BA01E960008137D
replicaID=7830
```

```
pipe="DS1 to DS2" class="DEFAULT" msg="Detected ADD of
uid=user.1001,ou=People,
dc=example,dc=com at ldap://server1.example.com:1389, but cannot create this
entry at the destination because an equivalent entry already exists at
ldap://server3.
example.com:3389. Details: Search using [search-criteria dn:
uid=user.1001,ou=People,
dc=example,dc=com attrsToGet: [*, dn]] returned results;
[uid=user.1001,ou=People,
dc=example,dc=com]. "
```

Perform the following steps to troubleshoot this type of problem:

Steps

1. Assuming that a possible DN mapping is ill-formed, first run the **`ldap-diff`** utility to compare the entries on the source and destination servers. Then look at the **`ldap-diff`** results with the mapping rules to determine shy the original search did not find a match.

   ```
   $ bin/ldap-diff \
     --outputLDIF config-difference.ldif \
     --baseDN "dc=example,dc=com" \
     --sourceHost server1.example.com \
     --targetHost server2.example.com \
     --sourcePort 1389 \
     --targetPort 3389 \
     --sourceBindDN "cn=Directory Manager" \
     --sourceBindPassword password \
     --searchFilter "(uid=1234)"
   ```

2. Review the destination server access logs to verify the search and filters used to find the entry. Typically, the key correlation attributes are not synchronized.
3. If the mapping rule attributes are not synchronized, review the Sync Classes and mapping rules, and use the information from the **`ldap-diff`** results to determine why a specific attribute may not be getting updated. Some questions to answer are as follows:
   - Is there more than one Sync Class that the operation could match?
   - If using an `include-base-dn` or `include-filter` in the mapping rules, does this exclude this operation by mistake?
   - If using an attribute map, are the mappings correct? Usually, the cause of this error is in the destination mapping attribute settings. For example, if a set of correlation attributes is defined as: `dn`, `mobile`, `accountNumber`, and the `accountNumber` changes for some reason, future operations on this entry will fail. To resolve this, either remove `accountNumber` from the rule, or add a second rule as: `dn,mobile`. The second rule is used only if the search using the first set of attributes fails. In this case, the entry is found and the `accountNumber` information is updated.
4. If deletes are being synchronized, check to see if there was a previous delete of this entry that was not synchronized properly. In some cases, simpler logic should be used for deletes due to the available attributes in the change logs. This scenario could cause an entry to not be deleted for some reason, which would cause an issue when a new entry with the same DN is added later. Use this information for mapping rules to see why the original search did not find a match.
5. Look at the destination directory server access logs to verify the search and filters it used to find the entry. Typically, the key attribute mappings are not synchronized.

Troubleshooting "No Match Found" failures

About this task

If there is a count for the No Match Found statistic using the **`status`** tool,verify the problem in the sync log. For example, if the **`status`** tool displays the following:

```
           --- Ops Completed for 'DS1 to DS2' Sync Pipe ---
Op Result              : Count
-----------------------:------
Success                : 0
Out Of Scope           : 0
Op Type Not Synced     : 0
No Change Needed       : 0
Entry Already Exists   : 1
No Match Found         : 1
Multiple Matches Found : 0
Failed During Mapping  : 0
Failed At Resource     : 0
Unexpected Exception   : 0
Total                  : 2
```

Verify the change in the `<server-root>/logs/sync` file to see the specific operation:

```
[12/May/2016:10:30:45 -0500] category=SYNC severity=MILD_WARNING
msgID=1893793952
op=4159648 changeNumber=6648922 replicationCSN=4beadaf4002f21150000
replicaID=8469-
ou=test,dc=example,dc=com pipe="DS1 to DS2" class="Others" msg="Detected
DELETE of
'uid=1234,ou=test,dc=example,dc=com' at ldap://server1.example.com:389, but
cannot
DELETE this entry at the destination because no matching entries were found
 at
ldap://
server2.example.com:389. Details: Search using [search-criteria dn:
uid=1234,ou=test,dc=alu,dc=com filter:
 (nsUniqueId=3a324c60-5ddb11df-80ffe681-
717b93af) attrsToGet: [*, accountNumber, dn, entryuuid, mobile, nsUniqueId,
object-
Class]] returned no results."
```

Perform the following steps to fix the issue:

Steps

1. Test the search using the filter in the error message, if displayed. For example, if the sync log specifies `filter: (nsUniqueId=3a324c60-5ddb11df-80ffe681-717b93af)`, use the **`ldapsearch`** tool to test the filter. If it is successful, is there anything in the attribute mappings that would exclude this from working properly?
2. Test the search using the full DN as the base. For example, use **`ldapsearch`** with the full DN `(uid=1234,ou=People,dc=example,dc=com)`. If it is successful, does the entry contain the attribute used in the mapping rule? If the attribute is not in the entry, determine if there is a reason why this value was not synchronized. Look at the attribute mappings and the filters used in the Sync Classes.

Troubleshooting "Failed at Resource" failures

About this task

If there is a count for the "Failed at Resource" statistic using the **`status`** tool, verify the problem in the sync log. For example, if the **`status`** tool displays the following information:

```
             --- Ops Completed for 'DS1 to DS2' Sync Pipe ---
Op Result               : Count
-----------------------:------
Success                 : 0
Out Of Scope            : 0
Op Type Not Synced      : 0
No Change Needed        : 0
Entry Already Exists    : 0
No Match Found          : 0
Multiple Matches Found : 0
Failed During Mapping   : 0
Failed At Resource      : 1
Unexpected Exception    : 0
Total :1
```

This will register after a change is detected at the source in any of the following cases:

- If the fetch of the full source entry fails. The entry exists but there is a connection failure, server down, timeout, or something similar.
- If the fetch of the destination entry fails or if the modification to the destination fails for an exceptional reason (but not for "Entry Already Exists," "Multiple Matches Found," or "No Match Found" issues).

Verify the change by viewing the `<server-root>/logs/sync` file to see the specific operation. If any of the following result codes are listed , the server is experiencing timeout errors:

- `resultCode=timeout: errorMessage=A client-side timeout was encountered while waiting 60000ms for a search response from server server1.example.com:1389`
- `resultCode=timeout: errorMessage=An I/O error occurred while trying to read the response from the server`
- `resultCode=server down: errorMessage=An I/O error occurred while trying to read the response from the server`
- `resultCode=server down: errorMessage=The connection to server server1.example.com:1389 was closed while waiting for a response to search request SearchRequest`
- `resultCode=object class violation: errorMessage='Entry device=1234,dc=example,dc=com violates the Directory Server schema configuration because it contains undefined object class`

With these "Failure at Destination" timeout errors, look at the following settings in the PingDirectory Server to determine if adjustments are needed:

Steps

1. For External Server Properties, check the connect-timeout property. This property specifies the maximum length of time to wait for a connection to be established before giving up and considering the server unavailable.
2. For the Sync Destination/Sync Source Properties, check the `response-timeout` property. This property specifies the maximum length of time that an operation should be allowed to be blocked while waiting for a response from the server. A value of zero indicates that there should be no client-side timeout. In this case, the server's default will be used.

```
$ bin/dsconfig --no-prompt --port 389 \
  --bindDN "cn=Directory Manager" \
```

```
  --bindPassword password list-external-servers \
  --property connect-timeout
```

```
External Server         : Type             : connect-timeout : response-
                                                               timeout
------------------------:------------------:-----------------:-----------
server1.example.com:389 : sundsee-ds       : 10 s            : -
server2.example.com:389 : sundsee-ds       : 10 s            : -
server3.example.com:389 : ping-identity-ds : 10 s            : -
server4.example.com:389 : ping-identity-ds : 10 s            : -
```

3. For Sync Pipe Properties, check the `max-operation-attempts`, `retry-backoff-initialwait`, `retry-backoff-max-wait`, `retry-backoff-increase-by`, `retry-backoff-percentage-increase`. These Sync Pipe properties provide tuning parameters that are used in conjunction with the timeout settings. When a Sync Pipe experiences an error, it will use these settings to determine how often and quickly it will retry the operation.

```
$ bin/dsconfig --no-prompt list-sync-pipes \
  --property max-operation-attempts --property retry-backoff-initial-wait
\
  --property retry-backoff-max-wait --property retry-backoff-increase-by \
  --property retry-backoff-percentage-increase \
  --port 389 --bindDN "cn=Directory Manager" \
  --bindPassword password
```

Installation and maintenance issues

The following are common installation and maintenance issues and possible solutions.

The setup program will not run

If the **`setup`** tool does not run properly, some of the most common reasons include the following:

A Java environment is not available

The server requires that Java be installed on the system prior to running the **`setup`** tool.

If there are multiple instances of Java on the server, run the **`setup`** tool with an explicitly defined value for the `JAVA_HOME` environment variable that specifies the path to the Java installation. For example:

```
$ env JAVA_HOME=/ds/java ./setup
```

Another issue may be that the value specified in the provided `JAVA_HOME` environment variable can be overridden by another environment variable. If that occurs, use the following command to override any other environment variables:

```
$ env UNBOUNDID_JAVA_HOME="/ds/java" UNBOUNDID_JAVA_BIN="" ./setup
```

Unexpected arguments provided to the JVM

If the **`setup`** tool attempts to launch the java command with an invalid set of arguments, it may prevent the JVM from starting. By default, no special options are provided to the JVM when running **`setup`**, but this might not be the case if either the `JAVA_ARGS` or `UNBOUNDID_JAVA_ARGS` environment variable is set. If the **`setup`** tool displays an error message that indicates that the Java environment could not be started with the provided set of arguments, run the following command:

```
$ unset JAVA_ARGS UNBOUNDID_JAVA_ARGS
```

The server has already been configured or started

The **setup** tool is only intended to provide the initial configuration for the server. It will not run if it detects that it has already been run.

A previous installation should be removed before installing a new one. However, if there is nothing of value in the existing installation, the following steps can be used to run the **setup** program:

- Remove the `config/config.ldiffile` and replace it with the `config/update/config.ldif.{revision}` file containing the initial configuration.
- If there are any files or subdirectories in the `db` directory, then remove them.
- If a `config/java.properties` file exists, then remove it.
- If a `lib/setup-java-home` script (or `lib\set-java-home.bat` file on Microsoft Windows) exists, then remove it.

The server will not start

If the server does not start, then there are a number of potential causes.

The server or other administrative tool is already running

Only a single instance of the server can run at any time from the same installation root. Other administrative operations can prevent the server from being started. In such cases, the attempt to start the server should fail with a message like:

```
The <server> could not acquire an exclusive lock on file
/ds/PingData<server>/locks/server.lock:
The exclusive lock requested for file
/ds/PingData<server>/locks/ server.lock
was not granted, which indicates that another
process already holds a shared or exclusive lock on
that file. This generally means that another instance
of this server is already running.
```

If the server is not running (and is not in the process of starting up or shutting down), and there are no other tools running that could prevent the server from being started, it is possible that a previously held lock was not properly released. Try removing all of the files in the locks directory before attempting to start the server.

There is not enough memory available

When the server is started, the JVM attempts to allocate all memory that it has been configured to use. If there is not enough free memory available on the system, the server generates an error message indicating that it could not be started.

There are a number of potential causes for this:

- If the amount of memory in the underlying system has changed, the server might need to be re-configured to use a smaller amount of memory.
- Another process on the system is consuming memory and there is not enough memory to start the server. Either terminate the other process, or reconfigure the server to use a smaller amount of memory.
- The server just shut down and an attempt was made to immediately restart it. If the server is configured to use a significant amount of memory, it can take a few seconds for all of the memory to be released back to the operating system. Run the **vmstat** Installation and maintenance issues command and wait until the amount of free memory stops growing before restarting the server.
- If the system is configured with one or more memory-backed file systems (such as `/tmp`), determine if any large files are consuming a significant amount of memory. If so, remove them or relocate them to a disk-based file system.

An invalid Java environment or JVM option was used

If an attempt to start the server fails with 'no valid Java environment could be found,' or 'the Java environment could not be started,' and memory is not the cause, other causes may include the following:

- The Java installation that was previously used to run the server no longer exists. Update the `config/java.properties` file to reference the new Java installation and run the `bin/dsjavaproperties` command to apply that change.
- The Java installation has been updated, and one or more of the options that had worked with the previous Java version no longer work. Re-configure the server to use the previous Java version, and investigate which options should be used with the new installation.
- If an `UNBOUNDID_JAVA_HOME` or `UNBOUNDID_JAVA_BIN` environment variable is set, its value may override the path to the Java installation used to run the server (defined in the `config/java.properties` file). Similarly, if an `UNBOUNDID_JAVA_ARGS` environment variable is set, then its value might override the arguments provided to the JVM. If this is the case, explicitly unset the `UNBOUNDID_JAVA_HOME`, `UNBOUNDID_JAVA_BIN`, and `UNBOUNDID_JAVA_ARGS`environment variables before starting the server.

Any time the `config/java.properties` file is updated, the `bin/dsjavaproperties` tool must be run to apply the new configuration. If a problem with the previous Java configuration prevents the `bin/dsjavaproperties` tool from running properly, remove the `lib/set-java-home` script (or `lib\set-java-home.bat` file on Microsoft Windows) and invoke the `bin/dsjavaproperties` tool with an explicitly-defined path to the Java environment, such as:

```
$ env UNBOUNDID_JAVA_HOME=/ds/java bin/dsjavaproperties
```

An invalid command-line option was used

There are a small number of arguments that can be provided when running the `bin/start-server` command. If arguments were provided and are not valid, the server displays an error message. Correct or remove the invalid argument and try to start the server again.

The server has an invalid configuration

If a change is made to the server configuration using **`dsconfig`** or the Administrative Console, the server will validate the change before applying it. However, it is possible that a configuration change can appear to be valid, but does not work as expected when the server is restarted.

In most cases, the server displays (and writes to the error log) a message that explains the problem. If the message does not provide enough information to identify the problem, the `logs/config-audit.logfile` provides recent configuration changes, or the `config/archived-configs` directory contains configuration changes not made through a supported configuration interface. The server can be started with the last valid configuration using the `-- useLastKnownGoodConfig` option:

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

To determine the set of configuration changes made to the server since the installation, use the **`config-difftool`** with the arguments `--sourceLocal --targetLocal --sourceBaseline`. The **`dsconfig --offline`** command can be used to make configuration changes.

Proper permissions are missing

The server should only be started by the user or role used to initially install the server. However, if the server was initially installed as a non-root user and then started by the root account, the server can no longer be started as a non-root user. Any new files that are created are owned by root.

If the user account used to run the server needs to change, change ownership of all files in the installation to that new user. For example, if the server should be run as the "ds" user in the "other" group, run the following command as root:

```
$ chown -R ds:other /ds/PingData<server>
```

The server has shutdown

Check the current server state by using the **bin/server-state** command. If the server was previously running but is no longer active, potential reasons may include:

- Shut down by an administrator – Unless the server was forcefully terminated, then messages are written to the error and server logs stating the reason.
- Shut down when the underlying system crashed or was rebooted – Run the **uptime** command on the underlying system to determine what was recently started or stopped.
- Process terminated by the underlying operating system – If this happens, a message is written to the system error log.
- Shut down in response to a serious problem – This can occur if the server has detected that the amount of usable disk space is critically low, or if errors have been encountered during processing that left the server without worker threads. Messages are written to the error and server logs (if disk space is available).
- JVM has crashed – If this happens, then the JVM should provide a fatal error log (a `hs_err_pid<processID>.log` file), and potentially a core file.

The server will not accept client connections

Check the current server state by using the **bin/server-state** command. If the server does not appear to be accepting connections from clients, reasons can include the following:

- The server is not running.
- The underlying system on which the server is installed is not running.
- The server is running, but is not reachable as a result of a network or firewall configuration problem. If that is the case, connection attempts should time out rather than be rejected.
- If the server is configured to allow secure communication through SSL or StartTLS, a problem with the key manager or trust manager configuration can cause connection rejections. Messages are written to the server access log for each failed connection attempt.
- The server may have reached its maximum number of allowed connections. Messages should be written to the server access log for each rejected connection attempt.
- If the server is configured to restrict access based on the address of the client, messages should be written to the server access log for each rejected connection attempt.
- If a connection handler encounters a significant error, it can stop listening for new requests. A message should be written to the server error log with information about the problem. Restarting the server can also solve the issue. Another option is to create an LDIF file that disables and then re-enables the connection handler, create the `config/auto-process-ldif` directory if it does not already exist, and then copy the LDIF file into it.

The server is unresponsive

Check the current server state by using the **bin/server-state** command. If the server process is running and appears to be accepting connections but does not respond to requests received on those connections, potential reasons for this include:

- If all worker threads are busy processing other client requests, new requests are forced to wait until a worker thread becomes available. A stack trace can be obtained using the **jstack** command to show the state of the worker threads and the waiting requests.

  If all worker threads are processing the same requests for a long time, the server sends an alert that it might be deadlocked. All threads might be tied up processing unindexed searches.

- If a request handler is busy with a client connection, other requests sent through that request handler are forced to wait until it is able to read data. If there is only one request handler, all connections are impacted. Stack traces obtained using the **`jstack`** command will show that a request handler thread is continuously blocked.
- If the JVM in which the server is running is not properly configured, it can spend too much time performing garbage collection. The effect on the server is similar to that of a network or firewall configuration problem. A stack trace obtained with the **`pstack`** utility will show that most threads are idle except the one performing garbage collection. It is also likely that a small number of CPUs is 100% busy while all other CPUs are idle. The server will also issue an alert after detecting a long JVM pause that will include details.
- If the JVM in which the server is running has hung, the **`pstack`** utility should show that one or more threads are blocked and unable to make progress. In such cases, the system CPUs should be mostly idle.
- If a there is a network or firewall configuration problem, communication attempts with the server will fail. A network sniffer will show that packets sent to the system are not receiving TCP acknowledgment.
- If the host system is hung or lost power with a graceful shutdown, the server will be unresponsive.

Problems with the Administrative Console

If a problem occurs when trying to use the Administrative Console, reasons might include one of the following:

- The web application container that hosts the console is not running. If an error occurs while trying to start it, consult the logs for the web application container.
- If a problem occurs while trying to authenticate, make sure that the target server is online. If it is, the access log might provide information about the authentication failure.
- If a problem occurs while interacting with the server instance using the Administrative Console, the access and error logs for that instance might provide additional information.

Problems with SSL communication

Enable TLS debugging in the server to troubleshoot SSL communication issues:

```
$ dsconfig create-debug-target \
  --publisher-name "File-Based Debug Logger" \
  --target-name
com.unboundid.directory.server.extensions.TLSConnectionSecurityProvider \
  --set debug-level:verbose \
  --set include-throwable-cause:true
```

```
$ dsconfig set-log-publisher-prop \
  --publisher-name "File-Based Debug Logger" \
  --set enabled:true \
  --set default-debug-level:disabled
```

In the `java.properties` file, add `-Djavax.net.debug=ssl` to the `start-ds` line, and run **`bin/dsjavaproperties`** to make the option take effect on a scheduled server restart.

Conditions for automatic server shutdown

All PingData servers will shutdown in an out of memory condition, a low disk space error state, or for running out of file descriptors. The PingDirectory Server will enter lockdown mode on unrecoverable database environment errors, but can be configured to shutdown instead with this setting:

```
$ dsconfig set-global-configuration-prop \
  --set unrecoverable-database-error-mode:initiate-server-shutdown
```

Insufficient memory errors

If the server shuts down due to insufficient memory errors, it is possible that the allocated heap size is not enough for the amount of data being returned. Consider increasing the heap size, or reducing the number of request handler threads using the following **dsconfig** command:

```
$ bin/dsconfig set-connection-handler-prop \
  --handler-name "HTTP Connection Handler" \
  --set num-request-handlers:<num-of-threads>
```

Enabling JVM debugging

About this task

Enable the JVM debugging options to track garbage collection data for the system. These options can impact JVM performance, but provide valuable data to tune the server. While the **jstat** utility with the `-gc` option can be used to obtain some information about garbage collection activity, there are additional arguments that can be added to provide additional detail, such as:

```
-XX:+PrintGCDetails
-XX:+PrintTenuringDistribution
-XX:+PrintGCApplicationConcurrentTime
-XX:+PrintGCApplicationStoppedTime
-XX:+PrintGCDateStamps
```

Steps

1. On the server, navigate to the `config/java.properties` file.
2. Edit the `config/java.properties` file. Add any additional arguments to the end of the line that begins with `start-<server>.java-args`.
3. Save the file.
4. Run the following command for the new arguments to take effect the next time the server is started:

```
$ bin/dsjavaproperties
```

Command-Line Tools

The PingDataSync Server provides a full suite of command-line tools necessary to administer the server. The command-line tools are available in the `bin` directory for UNIX or Linux systems and `bat` directory for Microsoft Windows systems.

Available command-line tools

PingDataSync Server provides the following command-line tools, which you can run in interactive, noninteractive, or script mode.

Tools Help

For	Use this option	Example
Information about arguments and subcommands Usage examples	`--help`	**dsconfig --help**

For	Use this option	Example
A list of subcommands	`--help-subcommands`	**`dsconfig --help-subcommands`**
More information about a subcommand	`--help` with the subcommand	**`dsconfig list-log-publishers --help`**

ⓘ **Note:** For detailed information and examples of the command-line tools, see the *Ping Identity PingDataSync Server Command-Line Tool Reference*.

Command-Line Tools

authrate	Perform repeated authentications against an LDAP directory server, where each authentication consists of a search to find a user followed by a bind to verify the credentials for that user.
backup	Run full or incremental backups on one or more Data Sync Server backends. This tool also supports the use of a properties file to pass predefined command-line arguments. See *Saving Options in a File* for more information.
base64	Encode raw data using the base64 algorithm or decode base64-encoded data back to its raw representation.
collect-support-data	Collect and package system information useful in troubleshooting problems. The information is packaged as a zip archive that can be sent to a technical support representative.
config-diff	Compare Data Sync Server configurations and produce a dsconfig batch file needed to bring the source inline with the target.
create-rc-script	Create a Run Control (RC) script to start, stop, and restart the Ping Identity Data Sync Server on UNIX-based systems.
create-sync-pipe-config	Create an initial Data Sync Server configuration.
create-systemd-script	Create a **`systemd`** script to start and stop the Ping Identity Data Sync Server on Linux-based systems.
deliver-one-time-password	Generate and deliver a one-time password to a user through some out-of-band mechanism. That password can then be used to authenticate via the UNBOUNDID-DELIVERED-OTP SASL mechanism.
deliver-password-reset-token	Generate and deliver a single-use token to a user through some out-of-band mechanism. The user can provide that token to the password modify extended request in lieu of the user's current password in order to select a new password.
dsconfig	View and edit the Data Sync Server configuration.
dsjavaproperties	Configure the JVM arguments used to run the PingDataSync Server and associated tools. Before launching the command, edit the properties file located in `config/java.properties` to specify the desired JVM options and JAVA_HOME environment variable.
dump-dns	Obtain a listing of all of the DNs for all entries below a specified base DN in the Data Sync Server.

encrypt-file	Encrypt or decrypt data using a key generated from a user-supplied passphrase, a key generated from an encryption settings definition, or a key shared among servers in the topology. The data to be processed can be read from a file or standard input, and the resulting data can be written to a file or standard output. You can use this command to encrypt and subsequently decrypt arbitrary data, or to decrypt encrypted backups, LDIF exports, and log files generated by the server.
encryption-settings	Manage the server encryption settings database.
enter-lockdown-mode	Request that the Data Sync Server enter lockdown mode, during which it only processes operations requested by users holding the `lockdown-mode` privilege.
export-reversible-passwords	Requests that the server export entries from a specified backend in LDIF form, including clear-text representations of any passwords encoded with a reversible storage scheme. This tool can only be used over a secure connection and when authenticated as a user with the permit-export-reversible-passwords privilege. The output is encrypted using a key generated from either a user-supplied passphrase or an encryption settings definition.
generate-totp-shared-secret	Generate a shared secret that you can use to generate time-based one-time password (TOTP) authentication codes for use in authenticating with the UNBOUNDID-TOTP SASL mechanism or with the validate TOTP password extended operation.
indent-ldap-filter	Parse a provided LDAP filter string and display it a multiline form that makes it easier to understand its hierarchy and embedded components. If possible, it might also simplify the provided filter in certain ways (for example, by removing unnecessary levels of hierarchy, like an AND embedded in an AND).
ldap-debugger	Intercept and decode LDAP communication.
ldap-diff	Compare the contents of two LDAP servers.
ldap-result-code	Display and query LDAP result codes.
ldapcompare	Perform compare operations in the PingDataSync Server. Compare operations can be used to efficiently determine whether a specified entry has a given value.
ldapdelete	Delete one or more entries from an LDAP directory server. You can provide the DNs of the entries to delete using named arguments, as trailing arguments, from a file, or from standard input. Alternatively, you can identify entries to delete using a search base DN and filter.
ldapmodify	Apply a set of add, delete, modify, and/or modify DN operations to a directory server. Supply the changes to apply in LDIF format, either from standard input or from a file specified with the 'ldifFile' argument. Change records must be separated by at least one blank line.
ldappasswordmodify	Update the password for a user in an LDAP directory server using the password modify extended operation (as defined in RFC 3062), a standard LDAP modify operation, or an Active Directory-specific modification.
ldapsearch	Process one or more searches in an LDAP directory server.

ldif-diff	Compare the contents of two files containing LDIF entries. The output will be an LDIF file containing the add, delete, and modify change records needed to convert the data in the source LDIF file into the data in the target LDIF file.
ldifmodify	Apply a set of changes (including add, delete, modify, and modify DN operations) to a set of entries contained in an LDIF file. The changes will be read from a second file (containing change records rather than entries), and the updated entries will be written to a third LDIF file. Unlike ldapmodify, the ldifmodify cannot read the changes to apply from standard input.
ldifsearch	Search one or more LDIF files to identify entries matching a given set of criteria.
leave-lockdown-mode	Request that the Data Sync Server leave lockdown mode and resume normal operation.
list-backends	List the backends and base DNs configured in Ping Identity Data Sync Server.
make-ldif	Generate LDIF data based on a definition in a template file. For example template files, see the server's `config/MakeLDIF` directory. In particular, the `examples-of-all-tags.template` file shows how to use all of the tags for generating values.
manage-account	Retrieve or update information about the current state of a user account. Processing is performed using the password policy state extended operation, and you must have the `password-reset` privilege to use this extended operation.
manage-certificates	Manage certificates and private keys in a JKS or PKCS #12 key store.
manage-extension	Install or update Ping Identity Data Sync Server extension bundles.
manage-profile	Generate, compare, install, and replace server profiles.
manage-tasks	Access information about pending, running, and completed tasks scheduled in the Data Sync Server.
manage-topology	Tool to manage the topology registry.
modrate	Perform repeated modifications against an LDAP directory server.
move-subtree	Move all entries in a specified subtree from one server to another.
parallel-update	Perform add, delete, modify, and modify DN operations concurrently using multiple threads.
prepare-endpoint-server	Prepare a Data Sync Server and an external server for communication.
profile-viewer	View information in data files captured by the Data Sync Server profiler.
realtime-sync	Control real-time synchronization including starting and stopping synchronization globally or for individual Sync Pipes; and setting the start point for real-time synchronization so that changes made before a specified time are ignored.
register-yubikey-otp-device	Register a YubiKey OTP device with the Directory Server for a specified user so that the device can be used to authenticate that user in conjunction with the UNBOUNDID-YUBIKEY-OTP SASL mechanism. Alternately, it can be used to deregister one or more YubiKey OTP devices for a user so that they can no longer be used to authenticate that user.

reload-http-connection-handler-certificates	Reload HTTPS Connection Handler certificates.
remove-backup	Safely remove a backup and optionally all of its dependent backups from the specified Data Sync Server backend.
remove-defunct-server	Remove a server from this server's topology.
replace-certificate	Replace the listener certificate for this Ping Identity Data Sync Server server instance.
restore	Restore a backup of a Data Sync Server backend.
resync	A resync operation can be used to resynchronize a Sync Destination with the contents of the Sync Pipe's corresponding Sync Source.
revert-update	Revert this server package's most recent update.
review-license	Review and/or indicate your acceptance of the license agreement defined in `legal/LICENSE.txt`.
rotate-log	Trigger the rotation of one or more log files.
sanitize-log	Sanitize the contents of a server log file to remove potentially sensitive information while still attempting to retain enough information to make it useful for diagnosing problems or understanding load patterns. The sanitization process operates on fields that consist of name-value pairs. The field name is always preserved, but field values might be tokenized or redacted if they might include sensitive information. Supported log file types include the file-based access, error, sync, and resync logs, as well as the operation timing access log and the detailed HTTP operation log. Sanitize the audit log using the scramble-ldif tool.
schedule-exec-task	Schedule an exec task to run a specified command in the server. To run an exec task, a number of conditions must be satisfied: the server's global configuration must have been updated to include 'com.unboundid.directory.server.tasks.ExecTask' in the set of allowed-task values, the requester must have the `exec-task` privilege, and the command to execute must be listed in the `exec-command-whitelist.txt` file in the server's `config` directory. The absolute path (on the server system) of the command to execute must be specified as the first unnamed trailing argument to this program, and the arguments to provide to that command (if any) should be specified as the remaining trailing arguments. The server root is used as the command's working directory, so any arguments that represent relative paths are interpreted as relative to that directory.
search-and-mod-rate	Perform repeated searches against an LDAP directory server and modify each entry returned.
search-logs	Search across log files to extract lines matching the provided patterns, like the **`grep`** command-line tool. The benefits of using this tool over **`grep`** are its ability to handle multiline log messages, extract log messages within a given time range, and the inclusion of rotated log files.
searchrate	Perform repeated searches against an LDAP directory server.
server-state	View information about the current state of the Data Sync Server process.
setup	Perform the initial setup for a server instance.
start-server	Start the Data Sync Server.

status	Display basic server information.
stop-server	Stop or restart the server.
subtree-accessibility	List or update the set of subtree accessibility restrictions defined in the Data Sync Server.
sum-file-sizes	Calculate the sum of the sizes for a set of files.
transform-ldif	Apply one or more changes to entries or change records read from an LDIF file, writing the updating records to a new file. This tool can apply a variety of transformations, including scrambling attribute values, redacting attribute values, excluding attributes or entries, replacing existing attributes, adding new attributes, renaming attributes, and moving entries from one subtree to another.
translate-ldif	Translates the contents of an LDIF file from the format for a Sync Source to the format of the Sync Destination using the filtering and mapping criteria defined for Sync Classes in the specified Sync Pipe.
uninstall	Uninstall Ping Identity Data Sync Server.
update	Update the PingDataSync Server to a newer version by downloading and unzipping the new server install package on the same host as the server you wish to update. Then, use the **`update`** tool from the new server package to update the older version of the server. Before upgrading a server, you should ensure that it is capable of starting without severe or fatal errors. During the update process, the server is stopped if running, then the update is performed, and a check is made to determine if the newly updated server starts without major errors. If it cannot start cleanly, the update will be backed out and the server returned to its prior state. See the **`revert-update`** tool for information on reverting an update.
validate-file-signature	Validate file signatures. For best results, file signatures should be validated by the same instance used to generate the file. However, it might be possible to validate signatures generated on other instances in a replicated topology.
validate-ldap-schema	Validate an LDAP schema read from one or more LDIF files.
validate-ldif	Validate the contents of an LDIF file against the server schema.
watch-entry	Launch a window to watch an LDAP entry for changes. If the entry changes, the background of modified attributes will temporarily be red. Attributes can be modified as well. This tool is primarily intended to demonstrate replication or synchronization functionality.

Saving Options in a File

Creating a tools properties file

You can set properties that apply to all tools or are tool-specific. These properties serve as defaults for the command-line options they represent.

Steps

1. Use a text editor to open the default tools properties file (`config/tools.properties`) or a different properties file.

   > ⓘ **Note:**
   >
   > If you use a file other than `config/tools.properties`, invoke the tool with the `--propertiesFilePath` option to specify the path to your properties file.

2. Set or change properties that apply to all tools.

   Use the standard Java properties file format (name=value) to set properties. For example, the following properties define a set of LDAP connection parameters.

   ```
   hostname=server1.example.com
   port=1389
   bindDN=cn=Directory\ Manager
   bindPassword=secret
   baseDN=dc=example,dc=com
   ```

   > ⓘ **Note:**
   >
   > Properties files do not allow quotation marks of any kind around values.
   >
   > Escape spaces and special characters.
   >
   > Whenever you specify a path, do not use `~` to refer to the home directory. The server does not expand the `~` value when read from a properties file.

3. Set or change properties that apply to specific tools.

   Tool-specific properties start with the name of the tool followed by a period. These properties take precedence over properties that apply to all tools. The following example sets two ports: one that applies to all tools (`port=1389`) and a tool-specific one that **`ldapsearch`** uses instead (`ldapsearch.port=2389`).

   ```
   hostname=server1.example.com
   port=1389
   ldapsearch.port=2389
   bindDN=cn=Directory\ Manager
   ```

4. Save your changes and close the file.

Evaluation order

PingDataSync Server uses the following evaluation ordering to determine options for a given command-line tool:

- All options on the tool's command line take precedence over any options in any properties file.
- If you use the `--propertiesFilePath` option with no other options, PingDataSync Server takes its options from the specified properties file.
- If you use no options on the command line including the `--propertiesFilePath` option (and `--noPropertiesFile`), PingDataSync Server searches for the `tools.properties` file at *<server-root>*.

- If no default properties file is found and a required option is missing, the tool generates an error.
- Tool-specific properties (for example, `ldapsearch.port=3389`) take precedence over general properties (for example, `port=1389`).

Consider this example properties file that is saved as *<server-root>*`/bin/tools.properties`:

```
hostname=server1.example.com
port=1389
bindDN=cn=Directory\ Manager
bindPassword=secret
```

PingDataSync Server locates a command-line option in a specific priority order.

1. All options presented with the tool on the command line take precedence over any options in any properties file. In the following example, the client request is run with the options specified on the command line (`--port` and `--baseDN`). The command uses the `bindDN` and `bindPassword` values specified in the properties file.

   ```
   $ bin/ldapsearch --port 2389 --baseDN ou=People,dc=example,dc=com \
     --propertiesFilePath bin/tools.properties "(objectclass=*)"
   ```
2. Next, if you specify the properties file using the `--propertiesFilePath` option and no other command-line options, PingDataSync Server uses the specified properties file as follows:

   ```
   $ bin/ldapsearch --propertiesFilePath bin/tools.properties \
     "(objectclass=*)"
   ```
3. If no options are presented with the tool on the command line and the `--noPropertiesFile` option is not present, PingDataSync Server attempts to locate any default `tools.properties` file in the following location:

   ```
   <server-root>/config/tools.properties
   ```

   If you move your `tools.properties` file from *<server-root>*`/bin` to the *<server-root>*`/config` directory. You can then run your tools as follows:

   ```
   $ bin/ldapsearch "(objectclass=*)"
   ```

   You can PingDataSync Server so that it does not search for a properties file by using the `--noPropertiesFile` option. This options tells PingDataSync Server to use only those options specified on the command line. You cannot use the `--propertiesFilePath` and `--noPropertiesFile` options together.
4. If no default `tools.properties` file is found and no options are specified with the command-line tool, then the tool generates an error for any missing arguments.

Sample dsconfig batch files

The `config/sample-dsconfig-batch-files` directory contains **`dsconfig`** batch files that you can use to configure various aspects of the server. For example, these files can enable additional security capabilities or take advantage of features that might require customization from one environment to another.

Each file includes comments that describe the purpose and benefit of its configuration change. You can choose which of the changes you want to apply.

You need to customize some of the batch files to provide values that might vary from one environment to another. To apply a batch file that requires changes, copy it to another directory and edit the copy. Leave the files in the `config/sample-dsconfig-batch-files` directory unchanged so that they can be updated when you upgrade the server. To specify the path to the file that contains the changes to apply, use the **`dsconfig`** tool (**`bin/dsconfig`** on UNIX-based systems or **`bat\dsconfig.bat`** on Windows) with the `--batch-file` argument.

You should also provide the arguments needed to connect and authenticate to the server. The `--no-prompt` argument ensures that the tool does not block while waiting for input if any necessary arguments are missing. Consider this example.

```
bin/dsconfig --hostname localhost \
  --port 636 --useSSL --trustStorePath config/truststore \
  --bindDN "uid=admin,dc=example,dc=com" \
  --bindPasswordFile admin-password.txt \
  --batch-file config/hardening-dsconfig-batch-files/reject-insecure-
request.dsconfig \
  --no-prompt
```

Sample dsconfig batch files

The `config/sample-dsconfig-batch-files` directory contains **`dsconfig`** batch files that you can use to configure various aspects of the server. For example, these files can enable additional security capabilities or take advantage of features that might require customization from one environment to another.

Each file includes comments that describe the purpose and benefit of its configuration change. You can choose which of the changes you want to apply.

You need to customize some of the batch files to provide values that might vary from one environment to another. To apply a batch file that requires changes, copy it to another directory and edit the copy. Leave the files in the `config/sample-dsconfig-batch-files` directory unchanged so that they can be updated when you upgrade the server. To specify the path to the file that contains the changes to apply, use the **`dsconfig`** tool (**`bin/dsconfig`** on UNIX-based systems or **`bat\dsconfig.bat`** on Windows) with the `--batch-file` argument.

You should also provide the arguments needed to connect and authenticate to the server. The `--no-prompt` argument ensures that the tool does not block while waiting for input if any necessary arguments are missing. Consider this example.

```
bin/dsconfig --hostname localhost \
  --port 636 --useSSL --trustStorePath config/truststore \
  --bindDN "uid=admin,dc=example,dc=com" \
  --bindPasswordFile admin-password.txt \
  --batch-file config/hardening-dsconfig-batch-files/reject-insecure-
request.dsconfig \
  --no-prompt
```

Running task-based tools

PingDataSync Server has a Tasks subsystem that allows you to schedule basic operations, such as **`backup`**, **`restore`**, **`rotate-log`**, **`schedule-exec-task`**, **`stop-server`**, and others. All task-based tools require the `--task` option that explicitly indicates the tool is to run as a task rather than in offline mode.

The following table shows the options that you can use for task-based operations:

Options for task-based operations

Option	Description
`--task`	Indicates that the tool is invoked as a task. The `--task` option is required. If you invoke a tool as a task without this `--task` option, then a warning message is displayed stating that it must be used. If the `--task` option is provided but the tool was not given the appropriate set of authentication arguments to the server, then an error message is displayed and the tool exits with an error.
`--start` *<startTime>*	Indicates the date and time, expressed in the format 'YYYYMMDDhhmmss', when the operation is to start. A value of '0' causes the task to be scheduled for immediate execution. After the scheduled run, the tool exits immediately.
`--dependency` *<taskID>*	Specifies the ID of a task upon which this task depends. A task does not start execution until all its dependencies have completed execution. You can use this option multiple times in a single command.
`--failedDependencyAction` *<action>*	Specifies the action this task takes if one of its dependent tasks fail. Valid action values are: ▪ `CANCEL` (the default) Cancels the task. ▪ `DISABLE` Disables the task so that it is not eligible to run until you manually enable it again. ▪ `PROCESS` Runs the task.
`--startAlert`	Generates an administrative alert when the task starts running.
`--errorAlert`	Generates an administrative alert when the task fails to complete successfully.
`--successAlert`	Generates an administrative alert when the task completes successfully.
`--startNotify` *<emailAddress>*	Specifies an email address to notify when the task starts running. You can use this option multiple times in a single command.
`--completionNotify` *<emailAddress>*	Specifies an email address to notify when the task completes, regardless of whether it succeeded or failed. You can use this option multiple times in a single command.

Option	Description
`--errorNotify <emailAddress>`	Specifies an email address to notify if an error occurs when this task executes. You can use this option multiple times in a single command.
`--successNotify <emailAddress>`	Specifies an email address to notify when this task completes successfully. You can use this option multiple times in a single command.

PingDataMetrics Server Administration Guide

Introduction

The PingDataMetrics Server collects performance data from the PingData Platform.

Topics include:

- PingDataMetrics Server Overview
- PingDataMetrics Server Components
- Data Collection
- Charts and Dashboards
- PostgreSQL DBS Details

PingDataMetrics Server overview

The PingDataMetrics Server provides insight into the transactions and performance of the Ping Data Platform.

The PingDataMetrics Server collects data from configured instances and replicas of the PingDirectory Server, the PingDirectoryProxy Server, the PingDataSync Server, and the PingDataGovernance Server. Data collected from the PingDataMetrics Server lets you perform the following actions:

- Measure the performance of the identity infrastructure as a whole service, not as a collection of individual servers.
- Identify client applications that require the greatest amount of resources.
- Determine which servers have the most available resources to handle requests.
- Predict the capacity and needs of the identity infrastructure to plan for increased traffic.
- Analyze all aspects of the identity infrastructure for troubleshooting performance issues.

PingDataMetrics Server components

The PingDataMetrics Server consists of several components, such as a server, REST API, query-metric tool, Simple Network Management Protocol (SNMP), a data set, and charts and dashboard templates.

The PingDataMetrics Server consists of the following components:

PingDataMetrics Server

A standalone server that relies on the PostgreSQL database for collected metrics. The PingDataMetrics Server gathers data for itself and configured PingData servers.

Metrics API

A REST API that provides access to collected metrics data. The API is accessible over HTTPS and supports multiple management parameters including filtering, averaging, and setting ranges for multiple data sets.

query-metric tool

The primary command-line tool for metric data access. This tool can also be used for scripted automation of extracting data from the PingDataMetrics Server. An explore option enables custom queries and additions to charts and dashboards.

SNMP access

Access system-level metrics over SNMP.

Data Set

A proprietary data structure that is designed for interoperability with charting libraries such as Highcharts, or Fusioncharts.

Charts, Chat Builder, and Dashboard templates

Tools for customizable, web-based metrics charts and dashboards.

Data collection

The PingDataMetrics Server collects data from all monitored servers through LDAP queries to the server's backend.

Each monitored server collects and stores a limited history of data locally. Data includes system status and performance information. To collect data, the PingDataMetrics Server regularly polls all monitored servers for data that is stored in time-contiguous blocks, gathers the recent data, and stores data in a PostgreSQL database. Polling has minimal impact on the monitored servers.

Performance data
The majority of information collected represents the performance of the monitored server.

Configure each monitored server to enable the PingDataMetrics Server to adequately keep up with the flow. Performance data represents multiple dimensions of a metric. For example, a response time metric can represent:

- The request type
- The time to respond
- The application that made the request
- The action that was taken

System and status data
All servers configured to be monitored by the PingDataMetrics Server store server and host system data.

Server and machine metrics are retrieved from the `cn=monitor` backend of the monitored server.

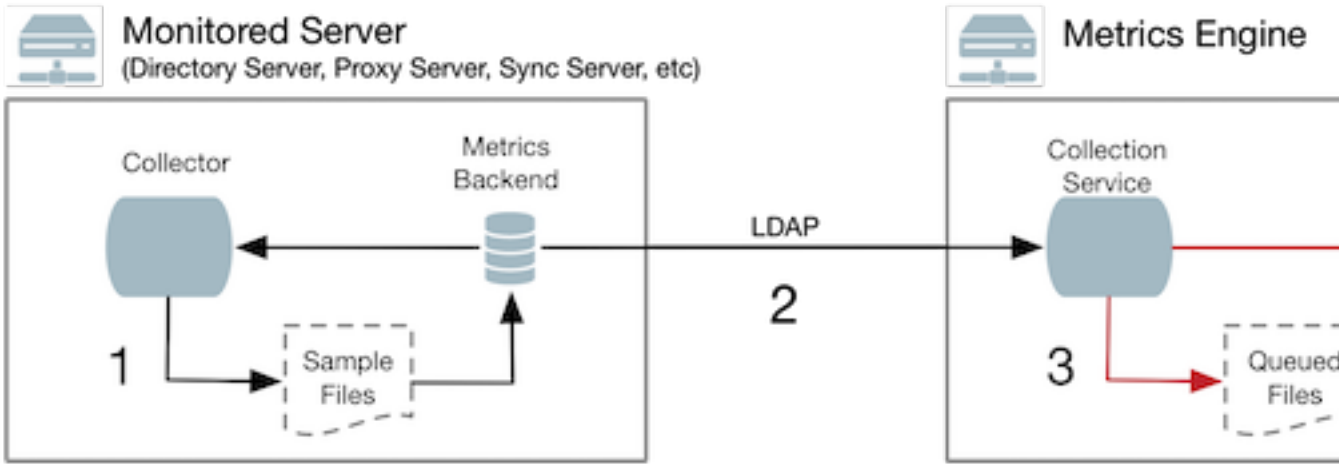
The Stats Collector plugin is responsible for collecting performance data from the `cn=monitor` backend. Data includes server responses, replication activity, local database activity, and host system metrics.

The Stats Collector configuration defines:

- Data sample and collection intervals
- The granularity of data collected, such as basic, extended, or verbose
- The types of host system data collected, such as CPU, disk, and network
- The type of data aggregation that occurs for LDAP application statistics

For more information, see Tuning Data Collection.

Data Collection



Processing steps

1. Data samples are taken and stored in time-contiguous blocks on the disk of the monitored server.
2. The PingDataMetrics Server collection service polls for new sample blocks.
3. The new sample blocks are queued to disk on the PingDataMetrics Server.
4. The PingDataMetrics Server import service loads new blocks into the database.

Charts and dashboards

The PingDataMetrics Server provides a number of charts and dashboards to display metrics information.

A Chart Builder tool enables configuring charts on an HTML page and saving the properties for use in a dashboard. Several charts are provided for general system information and specific PingData server functions. All dashboards are viewed from the PingDataMetrics Server.

PostgreSQL DBMS details

The PingDataMetrics Server uses a PostgreSQL DBMS to store data, which is included in the installation. This is a traditional, table-based DBMS, best suited for tabular data.

The PingDataMetrics Server interacts with the DBMS in four ways:

Data import

Import places steady write load on the DBMS and accounts for 80% of the writes. This single-threaded interaction puts a lock on the target table. A PingDataMetrics Server that monitors 20 servers keeps a single 10K RPM disk 70% busy with this single interaction.

Data aggregation

Data aggregation places a less frequent read/write load on the DBMS. This interaction is responsible for the aggregation of the data samples from one time resolution to the next, reading from one set of tables and writing to another set. Sample aggregation uses no table-level locks and the ratio of records between read:write is between 60:1 and 24:1.

Data sample age-out

Sample age-out occurs at regular intervals and results in a table being dropped or added. Age-out occurs every 30 minutes though some intervals might drop or add more than one table.

Data query

Sample queries occur when clients request metric samples from the public API. The API can aggregate multiple dimensions and multiple servers in a single request. A single request might fetch several million rows from the DBMS though it only returns a few hundred data points to the client. Samples from previous queries are cached by the PingDataMetrics Server, but initial queries for a given metric might take several seconds and result in a large amount of disk read activity

Over time, the storage of samples in the data tables is optimized to match the access patterns of the queries. However, the public API supports queries where the results are the aggregate of thousands of different dimension sets, and each dimension set might have thousands of samples within the time range of the query.

For example, a query about the throughput of all PingDirectory and PingDirectoryProxy Servers for all applications and all LDAP operations over the last 72 hours might result in four to six million DBMS records being read into memory, aggregated, and finally reduced to 100 data values. The results from each query are cached so that a subsequent request for the same data results in less DBMS activity. Both disk seek time and rotational delay impact the performance of a first-time query, so disks with faster RPM speeds provide a measurable improvement for first-time queries.

Installing the PingDataMetrics Server

Install and run the PingDataMetrics Server. It includes pre-installation requirements and considerations.

Topics include:

- Supported Platforms
- Installing Java
- Creating a Non-Root user
- Optimize the Linux Operating System
- Configuring the PingData Servers to Gather Metrics
- Ping License Keys
- Installing the PingDataMetrics Server
- Logging into the Administrative Console
- Server Folders and Files
- Adding Monitored Servers to the PingDataMetrics Server
- Removing Monitored Servers
- Start and Stop the PingDataMetrics Server Server
- Uninstall the PingDataMetrics Server
- Update Servers in a Topology
- Revert an update
- Administrative Accounts

Supported platforms

The following platforms and versions are supported for the 8.2 release.

Operating systems	Virtualization platforms	Java version
RedHat 6.6	VMWare vSphere & ESX 6.0	OpenJDK 8.x 64-bit
RedHat 6.8	KVM	OpenJDK 11.x 64-bit
RedHat 6.9	Amazon EC2	Oracle JDK 8.x 64-bit

Operating systems	Virtualization platforms	Java version
RedHat 7.4	Microsoft Azure (Supported by Professional Services)	Oracle JDK 11.x 64-bit
RedHat 7.5		
CentOS 6.8		
CentOS 6.9		
CentOS 7.4		
CentOS 7.5		
SUSE Enterprise 11 SP4		
SUSE Enterprise 12 SP3		
Ubuntu 16.04 LTS		
Ubuntu 18.04 LTS		
Amazon Linux		
Windows Server 2012 R2		
Windows Server 2016		

ⓘ **Note:**

You should have a Network Time Protocol (NTP) system in place so that multi-server environments are synchronized and timestamps are accurate.

Install the JDK

The Java 64-bit Java Development Kit (JDK) is required on the server.

Even if Java is already installed, create a separate Java installation for the server to use to ensure that updates to the system-wide Java installation do not inadvertently impact the installation.

Configure a non-root user

The PingDataMetrics Server installer cannot be run as the root user, and generally the PingDataMetrics Server (and PostgreSQL) should not be run as root.

ⓘ **Note:**

As a non-root user, network port numbers below 1024 cannot be used.

In general, the account must do the following:

- Listen on privileged network ports
- Bypass restrictions on resource limits

For security, the account should be restricted from the following:

- The ability to see processes owned by other users on the system.
- The ability to create hard links to files owned by other users on the system.

Optimize the Linux OS

Configure the Linux file system.

> ⓘ **Note:**
>
> The server explicitly overrides environment variables, like PATH, LD_LIBRARY_PATH, and LD_PRELOAD, to ensure that settings used to start the server do not inadvertently impact its behavior. If these variables must be edited, set values by editing the `set_environment_ vars` function of the `lib/_script-util.sh` script.
>
> Stop and restart the server for the change to take effect.

Setting the file descriptor limit

About this task

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

> ⓘ **Note:**
>
> If the operating system relies on `systemd`, see the Linux operating system documentation for instructions on setting the file descriptor limit.

Steps

1. Display the current hard limit of the system.

   The hard limit is the maximum server limit that can be set without tuning the kernel parameters in the `proc` filesystem.

   ```
   ulimit -aH
   ```

2. Edit the `/etc/sysctl.conf` file.
   - If the `fs.file-max` property is defined in the file, ensure the value is set to at least 65535.
   - If the line does not exist, add the following to the end of the file.

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

3. Edit the `/etc/security/limits.conf` file.
   - If the file has lines that set the soft and hard limits for the number of file descriptors, make sure the values are set to 65535.
   - If the lines are not present, add the following lines to the end of the file (before #End of file).

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

     > ⓘ **Note:**
     >
     > Insert a tab between the columns.

4. Reboot the system.
5. Run the `ulimit` command to verify that the file descriptor limit is set to 65535.

```
ulimit -n
```

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

ⓘ **Note:**

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

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

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

```
* soft nproc 65535
* soft nofile 65536
* hard nproc 65536
* hard nofile 65536
root soft nproc unlimited
```

Set the filesystem flushes
Linux systems running the ext3 filesystem only flush data to disk every five seconds.

If the server is on a Linux system, edit the mount options to include the following.

```
commit=1
```

This variable changes the flush frequency from fives seconds to one. Set the flush frequency in the `/etc/fstab` file to make sure the configuration remains after reboot.

Install sysstat and pstack on Red Hat

The server troubleshooting tool `collect-support-data` relies on the `iostat`, `mpstat`, and `pstack` utilities to collect monitoring, performance statistics, and stack trace information on the server's processes.

For Red Hat systems, make sure that these packages are installed.

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

Install the dstat utility

The `dstat` utility is used by the `collect-support-data` tool.

Disabling filesystem swapping
Disable any performance tuning services, such as `tuned`.

Steps

1. As root, change the current value in the operating system.
2. In `/etc/sysctl.conf`, add the line `vm.swappiness = 0`.
   This ensures the correct setting is applied when the system restarts.

3. If performance tuning is required, set `vm.swappiness` by cloning the existing performance profile, and then add `vm.swappiness = 0` to the new profile's `/usr/lib/tuned/profile-name/tuned.conf` file.
   a. To select the updated profile, run the command `tuned-adm profile customized_profile`.

Manage system entropy
Entropy is used to calculate random data that is used by the system in cryptographic operations.

Some environments with low entropy might have intermittent performance issues with SSL-based communication. This is more typical on virtual machines but can also occur in physical instances.

> ⓘ **Note:**
>
> For best results, monitor the `kernel.random.entropy_avail` in `sysctl` value.

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

Setting filesystem event monitoring (inotify)
Configure an event monitoring tool (such as **`inotify`**) for notifying processes about filesystem events, such as file creation, deletion, and updates.

About this task

The Linux system puts a limit on the number of inotify watches assigned to each user.

Steps

1. To increase the limit, edit `etc/systctl.conf` by adding the following line.

```
fs.inotify.max_user_watches = 524288
```

2. Run the following command.

```
$ sudo sysctl -w fs.inotify.max_user_watches=524288
```

Tuning the I/O scheduler
Use the correct I/O scheduler to increase performance and reduce the possibility of database timeouts when the system is under extreme write load.

About this task

For file systems running on an SSD or in a virtualized environment, use the `noop` scheduler. For all other systems, use the `deadline` scheduler.

Steps

1. To determine which scheduler is configured on your system, run the following command.

```
$ cat /sys/block/<block-device>/queue/scheduler
```

2. To change the scheduler on a running system, run the following command.

```
echo 'deadline' > /sys/block/<block-device>/queue/scheduler
```

3. To enable the change, restart the system.

> ⓘ **Note:**
>
> The procedure for configuring a scheduler to use at startup depends on the version of Linux. For the correct way to configure this setting, see the Linux documentation.

Enable the server to listen on privileged ports
Linux provides 'capabilities' used to grant specific commands the ability to do things that are normally reserved for a root account. Instead of granting the ability to a specific user, capabilities are granted to a specific command.

It might be convenient to enable the server to listen on privileged ports while running as a non-root user.

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

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

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

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

Run the following command for the change to take effect.

```
$ sudo ldconfig -v
```

Configure servers to be monitored

Before installing the PingDataMetrics Server, configure the servers to be monitored:

- PingDirectory Server
- PingDirectoryProxy Server
- PingDataSync Server
- PingDataGovernance Server

The monitored servers require sufficient disk space to store the monitoring data and can be configured with Tracked Applications if there are specific application bind distinguished names (DNs) that should be monitored.

Disk space requirements and monitoring intervals
The metrics backend on the monitored servers is responsible for the temporary storage of metric data and is configured to keep a maximum amount of metric history based on log retention policies, which are configured with the `dsconfig` tool.

The default retention policies define a cap on disk space usage, which in turn determines the amount of metric history retained. If the PingDataMetrics Server is stopped for a period of time, the monitored servers should be configured to retain enough metrics history to prevent gaps in data when the PingDataMetrics Server restarts. The amount of disk space required for metrics history might also depend on the monitored server's Stats Collector Plugin settings. In general, 500MB is enough to retain an eight-hour span of metrics history.

The value of the `sample-flush-interval property` of the monitored server's metrics backend determines the maximum delay between when a metric is captured and when it can be picked up by the PingDataMetrics Server. The flush interval can be set between 15 and 60 seconds, with longer values resulting in less processing load on the PingDataMetrics Server. However, this flush interval increases the latency between when the metric was captured and when it becomes visible in a chart or dashboard. Changing the `sample-flush-interval` attribute to 60 seconds has the PingDataMetrics Server keep 2000 minutes of history.

The number of metrics produced per unit of time varies based on the configuration. No formula can compute exact storage required for each hour of history. However, 60MB per hour is a standard estimate.

Tracked applications
If the PingDataMetrics Server monitors client applications associated with the monitored servers, configure the Tracked Applications feature for monitored servers as well.

Activity performed by a particular LDAP bind DN can be associated with a PingDataMetrics Server application-name, which in turn can be included in PingDataMetrics Server SLA definitions.

The Processing Time Histogram plugin is configured on each PingDirectory Server and PingDirectoryProxy Server as a set of histogram ranges. These ranges should be defined identically across all monitored servers. For each monitored server, set the `separate-monitor-entry-per-tracked-application` property of the processing time histogram plugin to `true`. Per-application monitoring information appears under `cn=monitor`. The `per-application-ldap-stats` property must also be set to `per-application-only` in the Stats Collector Plugin. For Tracked Application configuration details, see the Ping Identity PingDirectory Server Administration Guide.

The following example sets the required property of the Processing Time Histogram plugin.

```
$ bin/dsconfig set-plugin-prop \
  --plugin-name "Processing Time Histogram" \
  --set separate-monitor-entry-per-tracked-application:true
```

The following example sets the required property of the Stats Collector plugin.

```
$ bin/dsconfig set-plugin-prop \
  --plugin-name "Stats Collector" \
  --set per-application-ldap-stats:per-application-only
```

Ping license keys

License keys are required to install all Ping Products. Obtain license keys through Salesforce or from Ping Identity.

Important information about Ping license keys:

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

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

- Copy the license key file to the server root directory before running setup. The interactive setup tool discovers the file and not require input. If the file is not in the server root, the setup tool prompts for its location.

- If the license key is not in the server root directory, specify both the `--licenseKeyFile` option for non-interactive setup and the path to the file.

Installing the server

Use the `setup` tool to install the server.

About this task

The server must be started and stopped by the user who installed it.

> ⓘ **Note:**
>
> A Windows installation requires that the Visual Studio 2010 runtime patch be installed prior to running the setup command.

Steps

1. Sign on as a user other than root.
2. Obtain the latest `.zip` release bundle from Ping Identity and extract it in a directory owned by this user.

   ```
   $ unzip PingData<server-version>.zip
   ```
3. Change to the server root directory.

   ```
   $ cd PingData<server>
   ```
4. Run the `setup` command.

   ```
   $ ./setup
   ```
5. To accept the End-User License Agreement, enter `yes` or press **Enter** to accept the default.
6. Read the installation process and prerequisites. Press **Enter**.
7. Enter the port number of the PostgreSQL database instance to use to store monitoring or press **Enter** to accept the default port.
8. Enter the directory to be used for PostgreSQL data files or press **Enter** to accept the default.

   The default is `pgsql_data`.

   > ⓘ **Note:**
   >
   > If the name entered is a relative path name, it is created in the current working directory.
9. Enter a name for the database administrative account or press **Enter** to accept the default.

   > ⓘ **Note:**
   >
   > The `setup` tool creates a user (role) and database to be used by the PingDataMetrics Server. These credentials are strictly for use by this tool during this session and are not retained.
10. Enter and save a password.

11. Choose the name of the PostgreSQL account to be associated with the PingDataMetrics Server historical monitoring data or press **Enter** to accept the default.

    The default is `metrics`.

    > ⓘ **Note:**
    >
    > The tool creates this user account using the administrative account specified in step 9.

12. To enter and confirm a new password, type `yes` and provide a new password or press **Enter** to accept the default.
13. Enter the fully-qualified host name for the server or press **Enter** to accept the default.
14. Create the initial root user DN for the server or press **Enter** to accept the default.
15. Enter and confirm a password for this account.
16. Enter the port for HTTPS connection to the Platform APIs or press **Enter** to accept the default.

    The Platform APIs are the System for Cross-domain Identity Management (SCIM) and the Configuration.
17. Enter the port on which the PingDataMetrics Server accepts LDAP client connections or press **Enter** to accept the default.
18. To enable LDAPS, enter `yes` or press **Enter** to accept the default.

    The default is `no`.
19. If LDAPS is enabled, enter the port on which the server accepts LDAPS client connections or press **Enter** to accept the default.

    The default is `2636`.
20. To enable StartTLS, enter `yes` or press **Enter** to accept the default.

    The default is `no`.
21. Select a certificate option for the server.

    - Generate self-signed certificate. This is recommended for testing purposes only.
    - Use an existing certificate located on a Java KeyStore (JKS).
    - Use an existing certificate located on a PKCS12 KeyStore.
    - Use an existing certificate on a PKCS11 token.

    > ⓘ **Note:**
    >
    > Depending on the option you choose, you might need additional information. If you choose the Java or the PKCS#12 KeyStore, you need the KeyStore path and PIN. If you choose the PKCS#11 token, you need the key PIN.

22. Select the desired encryption for the directory data, backups, and log files.

    - Encrypt data with a key generated from an interactively provided passphrase. Using a passphrase (obtained interactively or read from a file) is the recommended approach for new deployments, and you should use the same encryption passphrase when setting up each server in the topology.
    - Encrypt data with a key generated from a passphrase read from a file.
    - Encrypt data with a randomly generated key. This is intended for testing, or if you intend to import the resulting encryption settings definition into other instances in the topology.
    - Do not encrypt server data.
23. Choose an option to assign the amount of memory that the server should allocate to the PingDataMetrics Server or press **Enter** to accept the default.
24. When the configuration is complete, press **Enter** (yes) to start the server.

25. To install the PingDataMetrics Server with the defined parameters, press **Enter**.

Next steps
After you install the PingDataMetrics Server server, access the Metrics landing page at https://*host*:*HTTPS-port*/view/index for access to the default dashboards, chart builder tool, and online documentation.

Log into the Administrative Console

After installing the server, access the Administrative Console at https://*host*/console/login to verify the configuration and manage the server.

The root user distinguished name (DN) or the common name of a root user DN is required to sign on to the Administrative Console. For example, if the DN created when the server was installed is `cn=Directory Manager`, directory manager can be used to sign on to the Administrative Console.

If the Administrative Console needs to run in an external container, such as Tomcat, you can install a separate package according to that container's documentation. Contact Ping Identity Customer Support for the package location and instructions.

Server folders and files

After extracting the distribution file, the following folders and command-line utilities are available.

Directories/Files/Tools	Description
ldif	Stores any LDIF files that you might have created or imported.
import-tmp	Store temporary imported items.
classes	Stores any external classes for server extensions.
db	For the PingDirectory Server, this is where its Berkeley DB files reside.
bak	Stores the physical backup files used with the backup command-line tool.
velocity	Stores Velocity templates that define the server's application pages.
update.bat, and update	The update tool for UNIX/Linux systems and Windows systems.
uninstall.bat. and uninstall	The uninstall tool for UNIX/Linux systems and Windows systems.
ping_logo.png	The image file for the Ping Identity logo.
setup.bat, and setup	The setup tool for UNIX/Linux systems and Windows systems.
revert-update.bat, and revert-update	The revert-update tool for UNIX/Linux systems and Windows systems.

Directories/Files/Tools	Description
README	README file that describes the steps to set up and start the server.
License.txt	Licensing agreement for the product.
legal-notices	Stores any legal notices for dependent software used with the product.
docs	Provides the release notes, Configuration Reference (HTML), API Reference, and all other product documentation.
metrics	Stores the metrics that can be gathered for this server and surfaced in the PingDataMetrics Server.
bin	Stores UNIX/Linux-based command-line tools.
bat	Stores Windows-based command-line tools.
lib	Stores any scripts, jar files, and library files needed for the server and its extensions.
collector	Used by the server to make monitored statistics available to the PingDataMetrics Server.
locks	Stores any lock files in the backends.
postgres	Stores PostgresSQL files.
tmp	Stores temporary files.
resource	Stores the MIB files for SNMP and can include ldif file, make-ldif templates, schema files, dsconfig batch files, and other items for configuring or managing the server.
config	Stores the configuration files for the backends (admin, config) as well as the directories for messages, schema, tools, and updates.
logs	Stores log files.

Add monitored servers to the PingDataMetrics Server

Configure the PingDataMetrics Server to monitor servers using the `monitored-servers` tool.

Using the monitored-servers tool

The monitored-servers command line tool configures communication between the servers and the PingDataMetrics Server, then adds external server definitions to the PingDataMetrics Server based on the server's administrative data.

Before a server is added to the PingDataMetrics Server configuration, the system determines whether communication needs configuring. If so, the `cn=Monitoring User` root user account is created on the external server.

Running the tool with the `add-servers` subcommand creates an external server based on the information discovered about the remote server. It also uses the information located in the `cn=admin data` entry to discover other servers in the topology, which are also added to the configuration. The following examples use the `monitored-servers` tool.

Run the monitored-servers tool with the add-servers subcommand. Specify connection information for the PingDataMetrics Server, as well as connection information for any remote servers in use.

```
$ bin/monitored-servers add-servers \
--bindDN uid=admin,dc=example,dc=com \
--bindPassword password \
--monitoringUserBindPassword password \
--remoteServerHostname localhost \
--remoteServerPort 1389 \
--remoteServerBindPassword password
```

Use the `--dry-run` option to generate output detailing the work that would be done in a live session without actually making changes to the server configuration.

```
$ bin/monitored-servers add-servers \
--bindDN uid=admin,dc=example,dc=com \
--bindPassword password \
--monitoringUserBindPassword password \
--remoteServerHostname localhost \
--remoteServerPort 1389 \
--remoteServerBindPassword password
--dry-run
```

Removing monitored servers

Remove monitored servers using the terminal.

About this task

Use the `monitored-servers` tool to remove servers from the PingDataMetrics Server.

Steps

1. To list the monitored servers, run the `/bin/status` tool.
2. From the monitored servers list, select a server.
3. To remove the desired server, run the following command.

```
bin/monitored-servers remove-server <Server-Name>
```

Start and stop the server

When the PingDataMetrics Server starts for the very first time, it downloads new samples from the monitored servers and adds data to the database. Until it has finished this first data collection, the PingDataMetrics Server cannot answer metric queries to the database. The PingDataMetrics Server processes samples from the oldest to the newest, so queries on more recent data might require more start-up time. If the monitored servers have been collecting samples for several days, there might be a significant backlog of data to collect.

To determine if the server is ready to respond to metric queries, run the `status` tool. If the `Sample Import Backlog` property is zero (0), the server is ready.

> ⓘ **Note:**
>
> The PingDataMetrics Server needs to be started and stopped by the user who installed it. If the PingDataMetrics Server is started or stopped by a different user, the following error is listed in the `postgres.log` file when Postgres starts.
>
> ```
> FATAL: role "<username>" does not exist
> ```

Starting the PingDataMetrics Server as a background process
Use the terminal to start the PingDataMetrics server as a background process.

Steps

1. Go to the server root directory.
2. Run the appropriate command for your system.
   - For Mac systems

     ```
     $ bin/start-server
     ```
   - For Windows systems

     ```
     $ bat/start-server
     ```

Starting the PingDataMetrics Server as a foreground process
Use the terminal to start the PingDataMetrics server as a foreground process.

Steps

1. Go to the server root directory.
2. Run the command `$ bin/start-server --nodetach`.

Starting the PingDataMetrics Server at boot time
Use the terminal to start the PingDataMetrics server at boot time.

Steps

1. Create the startup script as the non-root PingDataMetrics Server user.
   In the following example, `ds` is the user.

   ```
   $ bin/create-rc-script --outputFile Ping-Identity-ME.sh \
     --userName ds
   ```
2. Sign on as root.
3. Move the generated `Ping-Identity-ME.sh` script into the `/etc/init.d` directory, and create symlinks to it from the `/etc/rc3.d` directory (starting with an "S" to start the server) and the `/etc/rc0.d` directory (starting with a "K" to stop the server).

   ```
   # mv Ping-Identity-ME.sh /etc/init.d/
   # ln -s /etc/init.d/Ping-Identity-ME.sh /etc/rc3.d/S50-Ping-Identity-ME.sh
   # ln -s /etc/init.d/Ping-Identity-ME.sh /etc/rc0.d/K50-Ping-Identity-ME.sh
   ```

Stopping the PingDataMetrics Server
Use the terminal to stop the PingDataMetrics server.

Steps

1. Go to the server root directory.
2. Run the command `$ bin/stop-server`

Restarting the PingDataMetrics Server
Use the terminal to restart the PingDataMetrics Server using the `--restart` or `-R` option.

About this task

Running the restart command is equivalent to shutting down the server, exiting the Java virtual machine (JVM) session, and then starting up again, which requires a re-priming of the JVM cache.

Steps

1. Go to the server root directory.
2. Run the command `$ bin/stop-server --restart`.

Uninstalling the server

Use the `uninstall` command-line utility to uninstall the server using either interactive or non-interactive modes.

About this task

Interactive mode provides options, progress, and a list of the files and directories that must be manually deleted if necessary. Non-interactive mode, invoked with the `--no-prompt` option, suppresses progress information, except for fatal errors. All options for the uninstall command are listed with the `--help` option.

> ⓘ **Note:**
>
> Run the `uninstall` command as either the root user or the user account that installed the server.

Perform the following steps to uninstall in interactive mode.

Steps

1. Navigate to the server root directory.
   `$ cd PingData<server>`
2. To start the uninstall command, run `$ ./uninstall`.
3. Select the components to be removed, or press **Enter** to remove all components.

   > ⓘ **Note:**
   >
   > If the server is running, press **Enter** to shut down the server before continuing.
4. Manually remove any remaining files or directories.

Update servers in a topology

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

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

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

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

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

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

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

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

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

Updating the server

Before you begin

- An existing version of the server is stored at `PingData-server-old`.
- Make sure a complete, readable backup of the existing system is available before upgrading the server.
- There is a clear backout plan and schedule.

Steps

1. Download the latest version of the server software and unzip the file.
   For this example, the new server is located in the `PingData-server-new` directory.
2. Use the `update` tool of the newly unzipped build to update the server.
3. Specify the server instance that is being upgrading with the `--serverRoot` option.
4. To apply the update, stop the server.

Reverting an update
A server can be reverted to the previous version using the `revert-update` tool.

About this task

The `revert-update` tool accesses a log of file actions taken by the `update` tool to put the filesystem back to its prior state. If multiple updates have been run, the `revert-update` tool can be used multiple times to revert to each prior update sequentially. For example, the `revert-update` command can be run to revert to the server's previous state, then run again to return to its original state. The server is stopped during the revert-update process.

> ⓘ **Note:**
>
> Reverting an update is not supported for upgrades to version 7.0 due to the topology backend changes.

Steps

1. Navigate to the server root directory.
2. To revert back to the most recent version of the server, use the `revert-update` tool.

```
$ PingData-server-old/revert-update
```

Revert an update

After the server has been updated, you can revert one level back to the most recent version using the `revert-update` tool.

The `revert-update` tool accesses a log of file actions taken by the updater to put the filesystem back to its prior state. If you have run multiple updates, you can run the `revert-update` tool multiple times to revert to each prior update sequentially.

> ⓘ **Note:**
>
> You can only revert back one level. For example, if you have run the update twice since first installing the server, you can run the `revert-update` command to revert to its previous state, then run the `revert-update` command again to return to its original state.

Revert from version 7.x to a version prior to 7.0
Reverting from version 7.0 or later to a pre-7.0 version can be done using the `revert-update` command with some extra steps.

This is also the case when updating or reverting from a pre-6.2.0.2 version to 6.2.0.2 or later. These steps are listed when the `update` and `revert-update` tool are run as well. You might need to perform one or more of the following tasks, depending on your installation and configuration:

Processes for reverting from version 7.x to versions prior to 7.0

- When updating or reverting from 6.2.0.2 or later to a pre-6.2.0.2 version, indexes may need to be rebuilt. Older versions of the server use an incompatible format for Local DB Composite Indexes. To update a server with composite indexes in the previous format, delete these indexes and re-run the update. After the update is complete, recreate the indexes and use the rebuild-index tool to rebuild the indexes. The command for recreating an index will be in the "Undo" portion of the `logs/config-audit.log` file. If you wish to later revert to an older version, delete and recreate those composite indexes again after the revert has completed.
- When updating to 7.x for the first time, instance names will need to be set for each server in the topology if they were not previously set. This is done with the following `dsconfig` command:

  ```
  $ bin/dsconfig --bindDN "cn=Directory Manager" \
    --bindPassword secret \
    --no-prompt set-global-configuration-prop \
    --set instance-name:<name>
  ```

- Topology information such as server instances, instance and secret keys, server groups, and administrator users have moved to the topology portion of the configuration from the admin backend. As long as new servers are not added to the topology after this update, the `revert-update` command can be used to return to the previous version. However, if new servers are added, then the restored admin backend of this server will not contain information about the new servers, and the local server will not be able to communicate with any other servers in the topology. New servers should not be added to the topology if reverting this update is a possibility.
- If new servers were added to the topology after the update, the new servers must be temporarily removed from the topology until all servers have been reverted to the previous version.
- When a server is reverted to a pre-7.x version, any servers in the topology using the topology portion of the configuration (rather than the admin backend) will need to know that the reverted server was downgraded to the admin backend. This is done by running the following `dsconfig` command on one of the servers that has not been reverted:

  ```
  $ bin/dsconfig set-server-instance-prop \
    --instance-name <Reverted server instance name> \
    --set server-version:<Version to which server is reverted>
  ```

- If the topology does not have a master server when this command is run, it will not succeed. In this case, one of the remaining updated servers in the topology must be made master with the following command. This will enable the chosen instance to run the first command successfully.

  ```
  $ bin/dsconfig set-global-configuration-prop \
    --set force-as-master-for-mirrored-data:true
  ```

- The 7.x server version includes database changes that are not compatible with previous server versions (6.x or older). If you wish to later revert to an older version, the data must be exported to LDIF before performing the reversion. Re-import the data after the revert process has completed. In addition, the `changelogDb/` and `db/changelog/` directories in the reverted server root must be deleted after the revert has completed.

> ⓘ **Note:**
>
> When you start a server after a revert has been run, and the necessary extra steps have been completed, the server will display warnings about "offline configuration changes," but they are not critical and will not appear on subsequent startups.

Reverting to the latest server version
Revert your server to the previous, most recent version.

About this task

Steps

1. Navigate to the server root directory.
2. Use the `revert-update` tool to revert to the most recent version of the server.
   a. Run the command `$ <PingServer>-old/revert-update`

Administrative accounts

The `setup` tool automatically creates one administrative account when performing an installation.

Users that authenticate to the configuration API or the administrative console are stored in `cn=Root DNs,cn=config`. Accounts can be added or changed with the `dsconfig` tool.

Changing the administrative password
Change the administrative password in the terminal.

About this task

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

Steps

1. Open a text editor and create a text file containing the new password.
   `$ echo password > <new-password>.txt`
2. To change the root user's password, run `ldappasswordmodify`.

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

3. Remove the text file.
   `$ rm <new-password>.txt`

Managing the PingDataMetrics server

There are several ways to manage the PingDataMetrics Server status and performance.

The PingDataMetrics Server Administrative Console enables browser-based server management, the `dsconfig` tool enables command line management, and the Configuration API enables management by third-party interfaces.

PingDataMetrics server error logging

The PingDataMetrics Server provides logging for warnings, errors, or significant events that occur within the server. Log publishers rely on log rotation and retention policies.

Customization options for log publishers are available with the `dsconfig` command, (`bin/dsconfig` or `bat/dsconfig` on Windows).

Each log publisher must have at least one log rotation policy and log retention policy configured. Configure the log rotation policy for each log publisher. When a rotation limit is reached, the server rotates the current log and starts a new log.

Logging retention policies

There are several logging retention policies for PingDataMetrics server error logging.

Select retention configuration from the following:

Time limit rotation policy

Rotates the log based on the length of time since the last rotation. Default implementations are provided for rotation every 24 hours and every seven days.

Fixed time rotation policy

Rotates the logs every day at a specified time (based on 24-hour time). The default time is 2359.

Size limit rotation policy

Rotates the logs when the file reaches the maximum size for each log. The default size limit is 100MB.

Never rotate policy

Used in a rare event that does not require log rotation.

Logging rotation policies

There are different logging rotation configurations available for PingDataMetrics server error logging.

Select rotation configuration from the following:

File count retention policy

Sets the number of log files for the PingDataMetrics Server to retain. The default file count is 10 logs. If the file count is set to 1, then the log will continue to grow indefinitely without being rotated.

Free disk space retention policy

Sets the minimum amount of free disk space. The default free disk space is 500MB.

Size limit retention policy

Sets the maximum size of the combined archived logs. The default size limit is 500MB.

Custom retention policy

Create a new retention policy. This requires developing custom code to implement the log retention policy.

Never delete retention policy

Used in a rare event that does not require log deletion.

Creating log publishers

Create a new log publisher with `dsconfig`, either from the command line or in interactive mode.

About this task

Retention and rotation policies must be configured for the log publisher. Perform the following steps to create a log publisher.

> ⓘ **Note:**
>
> Compression cannot be disabled or turned off once configured for the logger. Determine logging requirements, prior to creating and configuring them.

Steps

1. Run the following command to create a log publisher with the `dsconfig` command that logs disconnect operations.

```
$ bin/dsconfig create-log-publisher \
  --type file-based-access --publisher-name "Disconnect Logger" \
  --set enabled:true \
  --set "rotation-policy:24 Hours Time Limit Rotation Policy" \
  --set "rotation-policy:Size Limit Rotation Policy" \
  --set "retention-policy:File Count Retention Policy" \
  --set log-connects:false \
  --set log-requests:false --set log-results:false \
  --set log-file:logs/disconnect.log
```

   a. To configure compression on the logger, add this option to the previous command:

```
--set compression-mechanism: gzip
```

2. To view log publishers, run the command:

```
$ bin/dsconfig list-log-publishers
```

Error log publisher
The Error Log reports errors, warnings, and informational messages about events that occur during the course of the server's operation.

Each entry in the error log records the following properties (some are disabled by default and must be enabled):

Time stamp

Displays the date and time of the operation in the format `DD/Month/YYYY:HH:MM:SS <offset from UTC time>.`

Category

Specifies the message category that is loosely based on the server components.

Severity

Specifies the message severity of the event, which defines the importance of the message in terms of major errors that need to be quickly addressed. The default severity levels are `fatal-error`, `notice`, `severe-error`, and `severe-warning`.

Message ID

Specifies the numeric identifier of the message.

Message

Stores the error, warning, or informational message.

ⓘ **Note:**

The following example displays an error log for the PingDataMetrics Server. The log is enabled by default and is accessible in the *<server-root>*`/logs/errors` file.

```
[21/Oct/2012:05:15:23.048 -0500] category=RUNTIME_INFORMATION
 severity=NOTICE
    msgID=20381715 msg="JVM Arguments: '-Xmx8g', '-Xms8g', '-
XX:MaxNewSize=1g',
```

```
    '-XX:NewSize=1g', '-XX:+UseConcMarkSweepGC', '-XX:
+CMSConcurrentMTEnabled',
    '-XX:+CMSParallelRemarkEnabled', '-XX:
+CMSParallelSurvivorRemarkEnabled',
    '-XX:+CMSScavengeBeforeRemark', '-XX:RefDiscoveryPolicy=1',
    '-XX:ParallelCMSThreads=4', '-
XX:CMSMaxAbortablePrecleanTime=3600000',
    '-XX:CMSInitiatingOccupancyFraction=80', '-XX:+UseParNewGC',
 '-XX:+UseMembar',
    '-XX:+UseBiasedLocking', '-XX:+UseLargePages', '-XX:
+UseCompressedOops',
    '-XX:PermSize=128M', '-XX:+HeapDumpOnOutOfMemoryError',
    '-Dcom.unboundid.directory.server.scriptName=setup'"
    [21/Oct/2012:05:15:23.081 -0500] category=EXTENSIONS
 severity=NOTICE
    msgID=1880555611 msg="Administrative alert type=server-
starting
    id=4178daee-ba3a-4be5-8e07-5ba17bf30b71
    class=com.unboundid.directory.server.core.MetricsEngine
    msg='The Metrics Server is starting'"
    [21/Oct/2012:05:15:23.585 -0500] category=CORE
 severity=NOTICE
    msgID=1879507338 msg="Starting group processing for backend
 api-users"
    [21/Oct/2012:05:15:23.586 -0500] category=CORE
 severity=NOTICE
    msgID=1879507339 msg="Completed group processing for backend
 api-users"
    [21/Oct/2012:05:15:23.586 -0500] category=EXTENSIONS
 severity=NOTICE
    msgID=1880555575 msg="'Group cache (2 static group(s) with 0
 total
    memberships and 0 unique members, 0 virtual static group(s),
    1 dynamic group(s))' currently consumes 7968 bytes and can
 grow to a maximum
    of an unknown number of bytes"
    [21/Oct/2012:05:16:18.011 -0500] category=CORE
 severity=NOTICE
    msgID=458887 msg="The Metrics Server (Metrics Server 4.5.1.0
    build 20121021003738Z, R12799) has started successfully"
```

ⓘ **Note:** Use `dsconfig` to modify the default File-Based Error Log, as in the following command:

```
$ bin/dsconfig set-log-publisher-prop \
     --publisher-name "File-Based Error Logger" \
     --set include-product-name:true --set include-instance-
name:true \
     --set include-startup-id:true
```

Configure log file encryption

The server supports the ability to encrypt log files as they are written.

The encrypt-log configuration property controls whether encryption will be enabled for the logger. Enabling encryption causes the log file to have an .encrypted extension (and if both encryption and compression are enabled, the extension will be `.gz.encrypted`). Any change that affects the name used for the log file could prevent older files from getting properly cleaned up.

Like compression, encryption can only be enabled when the logger is created. Encryption cannot be turned on or off once the logger is configured. For any log file that is encrypted, enabling compression is also recommended to reduce the amount of data that needs to be encrypted. This will also reduce the overall size of the log file. The `encrypt-file` tool (or custom code, using the LDAP SDK's `com.unboundid.util.PassphraseEncryptedInputStream`) is used to access the encrypted data.

To enable encryption, at least one encryption settings definition must be defined in the server. Use the one created during setup, or create a new one with the `encryption-settings` create command. By default, the encryption will be performed with the server's preferred encryption settings definition. To explicitly specify which definition should be used for the encryption, the `encryption-settings-definition-id` property can be set with the ID of that definition. It is recommended that the encryption settings definition is created from a passphrase so that the file can be decrypted by providing that passphrase, even if the original encryption settings definition is no longer available. A randomly generated encryption settings definition can also be created, but the log file can only be decrypted using a server instance that has that encryption settings definition.

When using encrypted logging, a small amount of data may remain in an in-memory buffer until the log file is closed. The encryption is performed using a block cipher, and it cannot write an incomplete block of data until the file is closed. This is not an issue for any log file that is not being actively written. To examine the contents of a log file that is being actively written, use the `rotate-log` tool to force the file to be rotated before attempting to examine it.

Setting log file encryption
Set log file encryption.

About this task

The following commands can be used to set log file encryption.

Steps

1. Use `dsconfig` to enable encryption for a Log Publisher.
   In this example, the FilebasedAccess Log Publisher "Encrypted Access" is created, compression is set, and rotation and retention policies are set.

```
$ bin/dsconfig create-log-publisher-prop --publisher-name "Encrypted
 Access" \
  --type file-based-access \
  --set enabled:true \
  --set compression-mechanism:gzip \
  --set encryption-settings-
definitionid:332C846EF0DCD1D5187C1592E4C74CAD33FC1E5FC20B726CD301CDD2B3FFBC2B
 \
  --set encrypt-log:true \
  --set log-file:logs/encrypted-access \
  --set "rotation-policy:24 Hours Time Limit Rotation Policy" \
  --set "rotation-policy:Size Limit Rotation Policy" \
  --set "retention-policy:File Count Retention Policy" \
  --set "retention-policy:Free Disk Space Retention Policy" \
  --set "retention-policy:Size Limit Retention Policy"
```

2. To decrypt and decompress the file:

```
$ bin/encrypt-file --decrypt \
  --decompress-input \
  --input-file logs/encrypted-access.20180216040332Z.gz.encrypted \
  --output-file decrypted-access
Initializing the server's encryption framework...DoneWriting decrypted
data to file '/ds/PingDirectory/decrypted-access' using akey generated
from encryption settings definition
'332c846ef0dcd1d5187c1592e4c74cad33fc1e5fc20b726cd301cdd2b3ffbc2b'Success
```

```
fully wrote 123,456,789 bytes of decrypted data
```

Backend monitor entries

Each PingData server exposes its monitoring information under the `cn=monitor` entry.

Administrators can use various means to monitor the servers through SNMP, the Administrative Console, JConsole, LDAP command-line tools, and the Stats Logger. The Monitor Backend contains an entry per component or activity being monitored. The list of all monitor entries can be seen using the ldapsearch command as follows:

```
$ bin/ldapsearch --hostname server1.example.com \
  --port 1389 \
  --bindDN "uid=admin,dc=example,dc=com" \
  --bindPassword secret \
  --baseDN "cn=monitor" "(objectclass=*)" cn
```

Monitoring Components

The following table lists a subset of monitor entries.

Component	Description
Active operations	number of active persistent searches.
Backends	Provides general information about the state of a server backend, including the entry count. If the backend is a local database, there is a corresponding database environment monitor entry with information on cache usage and on-disk size.
Client connections	Provides information about all client connections to the server including a name followed by an equal sign and a quoted value, such as connID="15", connectTime="20100308223038Z".
Connection handlers	Provides information about the available connection handlers on the server including the LDAP and LDIF connection handlers.
Disk space usage	Provides information about the disk space available to various components of the server.
General	Provides general information about the state of the server, including product name, vendor name, and server version.
Index	Provides information on each index including the number of preloaded keys and counters for read, write, remove, open-cursor, and read-for-search actions. These counters provide insight into how useful an index is for a given workload.
HTTP/HTTPS Connection Handler Statistics	Provides statistics about the interaction that the associated HTTP connection handler has had with its clients, including the number of connections accepted, average requests per connection, average connection duration, total bytes returned, and average processing time by status code.
JVM stack trace	Provides a stack trace of all threads processing within the JVM.

Component	Description
LDAP Connection Handler Statistics	Provides statistics about the interaction that the associated LDAP connection handler has had with its clients, including the number of connections established and closed, bytes read and written, LDAP messages read and written, and operations initiated, completed, and abandoned.
Processing time histogram	Categorizes operation processing times into a number of user-defined buckets of information, including the total number of operations processed, overall average response time (ms), and number of processing times between 0ms and 1ms.
System information	Provides general information about the system and the JVM on which the server is running, including system host name, operation system, JVM architecture, Java home, and Java version.
Version	Provides information about the server version, including build ID, and revision number.
Work queue	Provides information about the state of the server work queue, which holds requests until they can be processed by a worker thread, including the requests rejected, current work queue size, number of worker threads, and number of busy worker threads. The work queue configuration has a `monitor-queue-time` property set to `true` by default. This logs messages for new operations with a qtime attribute included in the log messages. Its value is expressed in milliseconds and represents the length of time that operations are held in the work queue.

Disk space usage monitor

The disk space usage monitor evaluates the free space at locations registered through the `DiskSpaceConsumer` interface.

The disk space usage monitor provides information about the amount of usable disk space available for server components. It also provides the ability to generate administrative alerts, as well as take action if the amount of usable space drops below the defined thresholds.

Disk space monitoring excludes disk locations that do not have server components registered. However, other disk locations may still impact server performance, such as the operating system disk, if it becomes full. When relevant to the server, these locations include the server root, the location of the `config` directory, the location of every log file, all JE backend directories, the location of the changelog, the location of the replication environment database, and the location of any server extension that registers itself with the `DiskSpaceConsumer` interface.

All values must be specified as absolute values or as percentages. A mix of absolute values and percentages cannot be used. The following thresholds are available:

Low space warning

This threshold defines either a percentage or an absolute amount of usable space. If the amount of usable space drops below this threshold, the server generates an administrative alert. It generates alerts at regular intervals, based on configuration settings, until the amount of usable space is increased, or as the amount of usable space is further reduced.

Low space error

This threshold is also defined as either a percentage or an absolute size. Once the amount of usable space drops below this threshold, the server will generate an alert notification and will begin rejecting all operations requested by nonroot users with "UNAVAILABLE" results. Once the server enters this mode, some action must be taken before the server will resume normal operations. This threshold must be less than or equal to the low space warning threshold. If they are equal, the server will begin rejecting requests from non-root users immediately upon detecting low usable disk space.

Out of space error

This threshold can also be defined as a percentage or an absolute size. Once the amount of usable space drops below this threshold, the server will generate a final administrative alert and will shut itself down. This threshold must be less than or equal to the low space error threshold. If they are equal, the server will shut itself down rather than rejecting requests from non-root users.

Notifications and alerts

Each PingData sever provides delivery mechanisms for account status notifications, administrative alerts, and alarms using SMTP, JMX, or SNMP

Alerts, alarms, and events reflect state changes within the server that may be of interest to a user or monitoring service. Account status notifications are only delivered to the account owner. Alert handler implementations include:

Error log alert handler

Sends administrative alerts to the configured server error logger(s).

Exec alert handler

Executes a specified command on the local system if an administrative alert matching the criteria for this alert handler is generated by the server. Information about the administrative alert is made available to the executed application as arguments provided by the command.

Groovy scripted alert handler

Provides alert handler implementations defined in a dynamically-loaded Groovy script that implements the `ScriptedAlertHandler` class defined in the Server SDK.

JMX alert handler

Sends administrative alerts to clients using the Java Management Extensions (JMX) protocol. PingData uses JMX for monitoring entries and requires that the JMX connection handler be enabled.

SMTP alert handler

Sends administrative alerts to clients via email using the SMTP. The server requires that one or more SMTP servers be defined in the global configuration.

SNMP alert handler

Sends administrative alerts to clients using the Simple Network Monitoring Protocol (SNMP). The server must have an SNMP agent capable of communicating through SNMP.

SNMP subagent alert handler

Sends SNMP traps to a master agent in response to administrative alerts generated within the server.

Third party alert handler

Provides alert handler implementations created in third-party code using the Server SDK.

ⓘ **Note:**

A complete listing of system alerts, alarms, and their severity is available in *<serverroot>*/docs/admin-alerts-list.csv

Configure alert handlers

You can configure alert handlers with the `dsconfig` tool.

PingData servers support JMX, SMTP, and SNMP. Use the --help option for a list of configuration options. The following is a sample command to create and enable an SMTP Alert handler from the command line:

```
$ bin/dsconfig create-alert-handler \
  --handler-name "SMTP Alert Handler" \
  --type smtp \
  --set enabled:true \
  --set "sender-address:alerts@example.com" \
  --set "recipient-address:administrators@example.com" \
  --set "message-subject:Directory Admin Alert \%\%alert-type\%\%" \
  --set "message-body:Administrative alert:\\n\%\%alert-message\%\%"
```

The alerts backend

PingData servers generate administrative alerts under the `cn=alerts` branch. The backend makes it possible to obtain admin alert information over LDAP for use with remote monitoring.

The backend's primary job is to process search operations for alerts. It does not support add, modify, or modify DN operations of entries. The alerts persist on disk in the `config/alerts.ldif` file so that they can survive server restarts. By default, the alerts remain on disk for seven days before being removed. However, administrators can configure the number of days for alert retention using the `dsconfig` tool. The administrative alerts of Warning level or worse that have occurred in the last 48 hours are viewable from the output of the `status` command-line tool and in the administrative console.

Viewing information in the alerts backend

View administrative alert information in the alert backend.

Steps

- Use `ldapsearch` to view the administrative alerts.

```
$ bin/ldapsearch --port 1389 --bindDN "cn=Directory Manager" \
  --bindPassword secret --baseDN cn=alerts "(objectclass=*)"
dn: cn=alerts
objectClass: top
objectClass: ds-alert-root
cn: alerts

dn: ds-alert-id=3d1857a2-e8cf-4e80-ac0e-ba933be59eca,cn=alerts
objectClass: top
objectClass: ds-admin-alert
ds-alert-id: 3d1857a2-e8cf-4e80-ac0e-ba933be59eca
ds-alert-type: server-started
ds-alert-severity: info
ds-alert-type-oid: 1.3.6.1.4.1.32473.2.11.33
ds-alert-time: 20110126041442.622Z
ds-alert-generator: com.unboundid.directory.server.core.metrics.engine
ds-alert-message: The server has started successfully
```

Modify the alert retention time

Use `dsconfig` to change the maximum time information about generated alerts retained in the alerts backend. After this time, the information is purged from the server.

> ⓘ **Note:**
>
> The minimum retention time is 0 milliseconds, which immediately purges the alert information.

This example shows a command to modify the alert retention time to 2 weeks.

```
$ bin/dsconfig set-backend-prop --backend-name "alerts" \
  --set "alert-retention-time: 2 weeks"
```

View the property using `dsconfig`:

```
$ bin/dsconfig get-backend-prop --backend-name "alerts" \
  --property alert-retention-time
```

Result:

```
Property : Value(s)
---------------------:---------
alert-retention-time : 2 w
```

Configure duplicate alert suppression

Configure duplicate alert suppression with `dsconfig`.

Use `dsconfig` to configure the maximum number of times an alert is generated within a particular time frame for the same condition. The `duplicate-alert-time-limit` property specifies the length of time that must pass before duplicate messages are sent over the administrative alert framework and the maximum number of messages should be sent.

```
$ bin/dsconfig set-global-configuration-prop \
  --set duplicate-alert-limit:2 \
  --set "duplicate-alert-time-limit:3 minutes"
```

System alarms, alerts, and gauges

Explanations and examples of the system alarms, alerts, and gauges in the PingData servers.

An alarm represents a stateful condition of the server or a resource that can indicate a problem, such as low disk space or external server unavailability. A gauge defines a set of threshold values with a specified severity that, when crossed, cause the server to enter or exit an alarm state. Gauges are used for monitoring continuous values like CPU load or free disk space (Numeric Gauge), or an enumerated set of values such as 'server available' or ‘server unavailable’ (Indicator Gauge). Gauges generate alarms, when the gauge’s severity changes due to changes in the monitored value. Like alerts, alarms have severity (NORMAL, WARNING, MINOR, MAJOR, CRITICAL), name, and message. Alarms will always have a Condition property, and may have a Specific Problem or Resource property. If surfaced through SNMP, a Probable Cause property and Alarm Type property are also listed. Alarms can be configured to generate alerts when the alarm's severity changes.

There are two alert types supported by the server - standard and alarm-specific. The server constantly monitors for conditions that might need attention by administrators, such as low disk space. For this condition, the standard alert is `low-disk-space-warning`, and the alarm-specific alert is `alarm-warning`. The server can be configured to generate alarm-specific alerts instead of, or in addition to, standard alerts. By default, standard alerts are generated for conditions internally monitored by the server. However, gauges can only generate alarm alerts.

The server installs a set of gauges that are specific to the product and that can be cloned or configured through the `dsconfig` tool. Existing gauges can be tailored to fit each environment by adjusting the update interval and threshold values. Configuration of system gauges determines the criteria by which alarms are triggered. The Stats Logger can be used to view historical information about the value and severity of all system gauges.

PingData servers are compliant with the International Telecommunication Union CCITT Recommendation X.733 (1992) standard for generating and clearing alarms. If configured, entering or exiting an alarm state can result in one or more alerts. An alarm state is exited when the condition no longer applies. An `alarm_cleared` alert type is generated by the system when an alarm's severity changes from a non-normal severity to any other severity. An `alarm_cleared` alert will correlate to a previous alarm when Condition and Resource property are the same. The Alarm Manager, which governs the actions performed when an alarm state is entered, is configurable through the `dsconfig` tool and administrative console.

Like the Alerts Backend, which stores information in `cn=alerts`, the Alarm Backend stores information within the `cn=alarms` backend. Unlike alerts, alarm thresholds have a state over time that can change in severity and be cleared when a monitored value returns to normal. Alarms can be viewed with the `status` tool. As with other alert types, alert handlers can be configured to manage the alerts generated by alarms. A complete listing of system alerts, alarms, and their severity is available in *<server-root>*`/docs/admin-alerts-list.csv`.

Testing alerts and alarms
Configure and test system alerts and alarms.

About this task

After configuring your alarms and alert handlers, verify that the server takes the appropriate action when an alarm state changes by manually increasing the severity of a gauge. Alarms and alerts can be verified with the `status` tool.

Steps

1. Configure a gauge with `dsconfig` and set the `override-severity` property to critical.
   The following example uses the CPU Usage (Percent) gauge.

```
$ dsconfig set-gauge-prop \
  --gauge-name "CPU Usage (Percent)" \
  --set override-severity:critical
```

2. Run the `status` tool to verify that an alarm was generated with corresponding alerts.

> ⓘ **Note:**
>
> The status tool provides a summary of the server's current state with key metrics and a list of recent alerts and alarms.

   The sample output has been shortened to show just the alarms and alerts information.

```
$ bin/status
```

```
--- Administrative Alerts ---
Severity : Time : Message
---------:----------------:-------------------------------------------
-------
Error : 11/Aug/2016 : Alarm [CPU Usage (Percent). Gauge CPU Usage
(Percent)
      : 15:41:00 -0500 : for Host System has
      : : a current value of '18.583333333333332'.
      : : The severity is currently OVERRIDDEN in the
```

```
      : : Gauge's configuration to 'CRITICAL'.
      : : The actual severity is: The severity is
      : : currently 'NORMAL', having assumed this
severity
      : : Mon Aug 11 15:41:00 CDT 2016. If CPU use is
high,
      : : check the server's current workload and make
any
      : : needed adjustments. Reducing the load on the
system
      : : will lead to better response times.
      : : Resource='Host System']
      : : raised with critical severity
Shown are alerts of severity [Info,Warning,Error,Fatal] from the past 48
hours
Use the --maxAlerts and/or --alertSeverity options to filter this list
      --- Alarms ---
Severity : Severity : Condition : Resource : Details
    : Start Time : : :
---------:------------:-----------:-------------:------------------------
--
Critical : 11/Aug/2016: CPU Usage : Host System : Gauge CPU Usage
(Percent) for
      : 15:41:00 : (Percent) : : Host System
      : -0500 : : : has a current value of
      : : : : '18.785714285714285'.
      : : : : The severity is
currently
      : : : : 'CRITICAL', having
assumed
      : : : : this severity Mon Aug 11
      : : : : 15:49:00 CDT 2016. If
CPU use
      : : : : is high, check the
server's
      : : : : current workload and
make any
      : : : : needed adjustments.
Reducing
      : : : : the load on the system
will
      : : : : lead to better response
times
Shown are alarms of severity [Warning,Minor,Major,Critical
Use the --alarmSeverity option to filter this list
```

Back up the PingDataMetrics Server database

The PingDataMetrics Server stores all historical metric samples in the PostgreSQL DBMS, along with several other data tables that are used for bookkeeping and normalization of the sample data.

Even a small PingDataMetrics Server installation, which monitors three to four servers, will use sample tables that occupy 95% of the total DBMS space. While a functional backup must capture a consistent view of several tables, the size of the sample tables dictates the desired approach to a regular backup strategy.

The historical samples enable:

- Diagnosing past performance problems
- Capacity planning and historical reporting
- Access to data needed for a revenue stream, such as data used for billing and charge back

Defining data that is important to the infrastructure will help determine the right backup strategy. In the case of billing, the data needed is typically small compared to the total population of the DBMS. This may be all the data needed, and the planning and resources required to backup the DBMS will be minimal.

> ⓘ **Note:**
>
> If it's not possible to determine what data will be important in the future, backing up all DBMS data is the safest approach.

Historical data storage
The PingDataMetrics Server DBMS stores all historical sample data.

The data in the PingDataMetrics Server DBMS is continually changing as long as the PingDataMetrics Server is running.

The system that feeds data to the PingDataMetrics Server is designed to allow the PingDataMetrics Server to be offline for hours at a time without dropping any data. The collection points hold the data for hours, giving the PingDataMetrics Server time for maintenance tasks. The collection points do have a limit on how long they hold data, so the PingDataMetrics Server cannot be offline for an indeterminate time.

If the PingDataMetrics Server is offline so long that the collection points start to delete data that has not yet been captured, then there will be gaps in the data. Aggregation still works, even with these gaps. If the data gap is four hours, four time samples will be missing in the one hour aggregation level, and no data will be missing in the one day aggregation level. However, the one day aggregation level will use only 20 hours of data rather than 24. By default, the PingDataMetrics Server can be offline for about eight hours before any data is lost.

The PingDataMetrics Server responds to queries that result in data with time gaps. The resulting data differentiates between data with zero value and missing data.

Planning the DBMS backup
Gather information to prepare the PingDataMetrics Server DBMS backup.

About this task

To prepare for backing up the PingDataMetrics DBMS, pick a time to backup, ensure you have enough space, and review additional data from sample deployments.

Steps

1. Choose a time window during which the PingDataMetrics Server can be offline.
2. Ensure there is enough disk space to hold the new image.

   The exact size of a DBMS table and its corresponding backup depends on the number of monitored servers, the number of tracked applications, the collected metrics, and the retention duration for each of the aggregation levels.

   The following table provides values from installations used during testing.

   Data from sample deployments

Data	25 Monitored Servers	50 Monitored Servers
1 minute data retention	14 days	14 days
1 hour data retention	52 weeks	52 weeks
1 day data retention	20 years	20 years

Data	25 Monitored Servers	50 Monitored Servers
1 second table size	22 G	42 G
1 minute table size	8 G	18 G
1 hour table size	4 G (estimated)	9 G (estimated)
1 day data retention	4 G (estimated)	7 G (estimated)
time to backup	15 minutes (estimated)	30 minutes (estimated)
time for import catchup	10 minutes	42 minutes
size of compressed backup image	3 G (estimated)	5.5 G (estimated)
time to restore	1 hour (estimated)	2 h (estimated)

ⓘ **Note:**

If no backups are performed and the DBMS is completely lost, reinitialize the DBMS, restart the PingDataMetrics Server, and start collecting data again. All collected metric and event data are lost, but the configuration required to start collecting data again is retained.

Starting the DBMS backup
Shut down the server and run the backup command.

Steps

1. Before a backup or restore, shut down the PingDataMetrics Server.
2. To backup the entire DMBS run the command:

```
$ tar -cf backup.tar <path-to-postgres-data-directory>
```

Restoring a DBMS backup
Fully restore a DBMS backup to a new database.

Steps

- To restore the full backup to a new database, run the command from the base directory of the PostgreSQL data directory:

  ```
  $ tar -xvf backup.tar
  ```

  For more information, the PostgreSQL website is a useful resource.

Management tools

The PingDataMetrics Server provides several command-line tools to administer the server.

The command-line tools are available in the `bin` directory for UNIX or Linux systems and `bat` directory for Microsoft Windows systems.

Each command-line utility provides a description of the subcommands, arguments, and usage examples needed to run the tool.

Subcommands

- View detailed argument options and examples by typing `--help` with the command:

```
$ bin/dsconfig --help
```

- To list the subcommands for each command:

```
$ bin/dsconfig --help-subcommands
```

- To list more detailed subcommand information:

```
$ bin/dsconfig list-log-publishers --help
```

Available command-line utilities
Review a list of available command-line tools and their descriptions.

The following command-line utilities are available, which can be run in interactive, noninteractive, or script mode.

Command Line Tools

Command-Line Tool	Description
`backup`	Run full or incremental backups on one or more PingDataMetrics Server backends. This utility also supports the use of a properties file to pass predefined command-line arguments. See Managing the tools.properties File for more information.
`base64`	Encode raw data using the base64 algorithm or decode base64-encoded data back to its raw representation.
`collect-support-data`	Collect and package system information useful in troubleshooting problems. The information is packaged as a `.zip` archive that can be sent to a technical support representative.
`create-rc-script`	Create an Run Control (RC) script that may be used to start, stop, and restart the server on UNIX-based systems.
`config-diff`	Generate a summary of the configuration changes in a local or remote server instance. The tool can be used to compare configuration settings when troubleshooting issues, or when verifying configuration settings on new servers.
`dsconfig`	View and edit the server configuration.
`dsframework`	Manage administrative server groups or the global administrative user accounts that are used to configure servers within server groups.
`dsjavaproperties`	Configure the Java virtual machine (JVM) arguments used to run the server and associated tools. Before launching the command, edit the properties file located in `config/java.properties` to specify the desired JVM options and `JAVA_HOME` environment variable.

Command-Line Tool	Description
`ldapmodify`	Perform LDAP modify, add, delete, and modify DN operations.
`ldappasswordmodify`	Perform LDAP password modify operations.
`ldapsearch`	Perform LDAP search operations.
`ldif-diff`	Compare the contents of two LDIF files, the output being an LDIF file needed to bring.
`ldifmodify`	Apply a set of modify, add, and delete operations against data in an LDIF file.
`manage-extension`	Install or update extension bundles. An extension bundle is a package of extension(s) that utilize the Server SDK to extend the functionality of the server. Extension bundles are installed from a `.zip` archive or file system directory. The server is restarted to activate the extension(s).
`metric-engine-schema`	Show current and required DBMS schema version information.
`monitored-servers`	Configure the set of servers to be monitored by this PingDataMetrics Server and prepare external servers for monitoring.
`query-metric`	Explore collected monitoring data by forming queries for data.
`queryrate`	Execute metric queries.
`restore`	Restore a backup of the server backend.
`revert-update`	Returns a server to the version before the last update was performed.
`review-license`	Review or accept the product license.
`server-state`	View information about the current state of the server process.
`setup`	Perform the initial setup for the server instance.
`start-server`	Start the server
`status`	Display basic server information.
`stop-server`	Stop or restart the server.
`sum-file-sizes`	Calculate the sum of the sizes for a set of files.
`uninstall`	Uninstall the server

Command-Line Tool	Description
`update`	Update the server to a newer version by downloading and unzipping the new server install package on the same host as the server to update. Use the update tool from the new server package to update the older version of the server. During the update process, the server is stopped if running, then the update is performed. A check is performed to determine if the newly updated server starts without major errors. If it cannot start cleanly, the update is backed out and the server is returned to its prior state. See the `revert-update` tool for information on reverting an update.

The tools.property file

The `tools.properties` file simplifies command-line invocations by reading in a set of arguments for each tool from a text file. Each property consists of a name/value pair for a tool's arguments.

Two types of properties files are supported:

- Default properties files that can be applied to all command-line utilities.
- Tool-specific properties file that can be specified using the `--propertiesFilePath` option.

> ⓘ **Note:**
>
> All of the server's command-line utilities can be overwritten using the `config/tools.properties` file.

Create a properties file with a text editor or using the standard Java properties file format (name=value). For example, create a simple properties file that defines a set of LDAP connection parameters as follows:

```
hostname=server1.example.com
port=1389
bindDN=cn=Directory\ Manager
bindPassword=secret
```

Specify the location of the file using the `--propertiesFilePath` option. For example, specify the path to the properties file with `ldapsearch` as follows:

```
$ bin/ldapsearch --propertiesFilePath bin/mytools.properties
 "(objectclass=*)"
```

> ⓘ **Note:**
>
> Properties files do not allow quotation marks around values. Any spaces or special characters should be removed.

Tool-specific properties

The server also supports properties for specific tool options using the format: `tool.option=value`.

Tool-specific options have precedence over general options. For example, the following properties file uses `ldapsearch.port=2389` for `ldapsearch` requests by the client. All other tools that use the properties file use `port=1389`.

Example using `port=1389`:

```
hostname=server1.example.com
port=1389
ldapsearch.port=2389
```

```
bindDN=cn=Directory\ Manager
```

Example using the `dsconfig` configuration tool:

```
hostname=server1.example.com
port=1389
bindDN=cn=Directory\ Manager
dsconfig.bindPasswordFile=/ds/config/password
```

Specify default properties files
The server provides a default properties file, `tools.properties`, that applies to all command-line utilities used in client requests.

To use a file with a different filename in the default `tools.properties` location, specify the path using the `--propertiesFilePath` option.

Evaluation order
Evaluation ordering determines which tools, options, or properties take precedence over others in the command-line utility.

The following evaluation ordering is used to determine options for a given command-line utility:

- All options used with a utility on the command line take precedence over any options in any properties file.
- If the `--propertiesFilePath` option is used with no other options, the server takes its options from the specified properties file.
- If no options are used on the command line including the `--propertiesFilePath` option (and `--noPropertiesFile`), the server searches for the `tools.properties` file at *<server-root>*.
- If no default properties file is found and a required option is missing, the tool generates an error.
- Tool-specific properties (such as `ldapsearch.port=3389`) have precedence over general properties (such as `port=1389`).

HTTP Connection Handlers

HTTP Connection Handlers are responsible for managing the communication with HTTP clients and invoking servlets to process requests from those clients, as well as host web applications on the server.

Each HTTP connection handler must be configured with one or more HTTP servlet extensions and zero or more HTTP operation log publishers.

If you cannot start the HTTP Connection Handler (for example, if its associated HTTP Servlet Extension fails to initialize), then this will not prevent the entire Directory Proxy Server from starting. The server's start-server tool will output any errors to the error log. This allows the server to continue serving LDAP requests even with a bad servlet extension. The configuration properties available for use with an HTTP connection handler include:

listen-address

Specifies the address on which the connection handler will listen for requests from clients. If not specified, then requests will be accepted on all addresses bound to the system.

listen-port

Specifies the port on which the connection handler will listen for requests from clients. This is required.

use-ssl

Indicates whether the connection handler will use SSL/TLS to secure communications with clients (whether it uses HTTPS rather than HTTP). If SSL is enabled, then `key-manager-provider` and `trust-manager-provider` values must also be specified.

http-servlet-extension

Specifies the set of servlet extensions that will be enabled for use with the connection handler. You can have multiple HTTP connection handlers (listening on different address/port combinations) with identical or different sets of servlet extensions. At least one servlet extension must be configured.

http-operation-log-publisher

Specifies the set of HTTP operation log publishers that should be used with the connection handler. By default, no HTTP operation log publishers will be used.

key-manager-provider

Specifies the key manager provider that will be used to obtain the certificate presented to clients if SSL is enabled.

trust-manager-provider

Specifies the trust manager provider that will be used to determine whether to accept any client certificates presented to the server.

num-request-handlers

Specifies the number of threads that should be used to process requests from HTTP clients. These threads are separate from the worker threads used to process other kinds of requests. The default value of zero means the number of threads will be automatically selected based on the number of CPUs available to the Java virtual machine (JVM).

web-application-extension

Specifies the Web applications to be hosted by the server.

Configuring an HTTP connection handler

Configure an HTTP connection handler by configuring the HTTP servlet extensions and the HTTP log publishers.

About this task

An HTTP connection handler has two dependent configuration objects: one or more HTTP servlet extensions and optionally, an HTTP log publisher. The HTTP servlet extension and log publisher must be configured prior to configuring the HTTP connection handler. The log publisher is optional but in most cases, you want to configure one or more logs to troubleshoot any issues with your HTTP connection.

Steps

1. Configure your HTTP servlet extensions.

   The following example uses the ExampleHTTPServletExtension in the Server SDK.

```
$ bin/dsconfig create-http-servlet-extension \
  --extension-name "Hello World Servlet" \
  --type third-party \
  --set
"extensionclass:com.unboundid.directory.sdk.examples.ExampleHTTPServletEx
  tension" \
  --set "extension-argument:path=/" \
  --set "extension-argument:name=example-servlet"
```

2. Configure one or more HTTP log publishers.

   The following example configures two log publishers: one for common access; the other, detailed access. Both log publishers use the default configuration settings for log rotation and retention.

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

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

3. Configure the HTTP connection handler by specifying the HTTP servlet extension and log publishers.

> ⓘ **Note:**
>
> Some configuration properties can be later updated on the fly while others, such as listen-port, require that the HTTP Connection Handler be disabled, then re-enabled for the change to take effect.

```
$ bin/dsconfig create-connection-handler \
  --handler-name "Hello World HTTP Connection Handler" \
  --type http \
  --set enabled:true \
  --set listen-port:8443 \
  --set use-ssl:true \
  --set "http-servlet-extension:Hello World Servlet" \
  --set "http-operation-log-publisher:HTTP Common Access Logger" \
  --set "http-operation-log-publisher:HTTP Detailed Access Logger" \
  --set "key-manager-provider:JKS" \
  --set "trust-manager-provider:JKS"
```

4. Monitor the connection handler using the `ldapsearch` tool.

> ⓘ **Note:**
>
> The HTTP Connection Handler has an advanced monitor entry property, `keep-stats`, that is set to TRUE by default.

```
$ bin/ldapsearch --baseDN "cn=monitor" \
  "(objectClass=ds-http-connection-handler-statistics-monitor-entry)"
```

HTTP correlation IDs

The correlation ID allows related log messages to be easily located and grouped.

A typical request to a software system is handled by multiple subsystems, many of which can be distinct servers residing on distinct hosts and locations. Tracing the request flow on distributed systems can be

challenging, as log messages are scattered across various systems and intermingled with messages for other requests. To make this easier, a correlation ID can be assigned to a request, which is then added to every associated operation as the request flows through the larger system. The server supports correlation IDs for all HTTP requests received through its HTTP(S) Connection Handler.

When an HTTP request is received, it is automatically assigned a correlation ID. This ID can be used to correlate HTTP responses with messages recorded to the HTTP Detailed Operation log and the trace log. For specific web APIs, the correlation ID may also be passed to the LDAP subsystem. For the SCIM 1, Delegated Admin, Consent, and Directory REST APIs, the correlation ID will also appear with associated requests in LDAP logs in the `correlationID` key. The correlation ID is also used as the default client request ID value in Intermediate Client Request Controls used by the SCIM 2, Consent, and Directory REST APIs. Values related to the Intermediate Client Request Control appear in the LDAP logs in the `via` key, and are forwarded to downstream LDAP servers when received by the PingDirectoryProxy server. The correlation ID header is also added to requests forwarded by the PingDataGovernance gateway.

For Server SDK extensions that have access to the current HttpServletRequest, the current correlation ID can be retrieved as a string through the `HttpServletRequest's com.pingidentity.pingdata.correlation_id` attribute.

```
(String)
 request.getAttribute("com.pingidentity.pingdata.correlation_id");
```

Configuring HTTP correlation ID support
Enable or disable correlation ID support for the HTTPS Connection Handler.

About this task

Correlation ID support is enabled by default for each HTTP Connection Handler.

Steps

- To enable correlation ID support for the HTTPS Connection Handler, run the command:

```
$ bin/dsconfig set-connection-handler-prop \
  --handler-name "HTTPS Connection Handler" \
  --set use-correlation-id-header:true
```

- To disable the correlation ID support for the HTTPS Connection Handler, run the commands:

```
$ bin/dsconfig set-connection-handler-prop \
  --handler-name "HTTPS Connection Handler" \
  --set use-correlation-id-header:false
```

Configure the correlation ID response header
Configure the server's correlation ID response header name.

The server will generate a correlation ID for every HTTP request and send it in the response through the `Correlation-Id` response header. This response header name can be customized.

The following example changes the `correlation-id-response-header` property to "X-Request-Id."

```
$ bin/dsconfig set-connection-handler-prop \
  --handler-name "HTTPS Connection Handler" \
  --set correlation-id-response-header:X-Request-Id
```

Accept an incoming correlation ID from the request
Configure your HTTP request headers to accept an incoming correlation ID.

About this task

By default, the server generates a new, unique correlation ID for each HTTP request, and ignores any correlation ID that can be set on the request. This can be changed by designating the names of one or more HTTP request headers that contain an existing correlation ID value. This enables the server to integrate with a larger system consisting of every server using correlation IDs.

Steps

- To accept an incoming correlation ID from the request, designate the names of multiple HTTP request headers that contain an existing correlation ID value.

Example

```
$ bin/dsconfig set-connection-handler-prop \
  --handler-name "HTTPS Connection Handler" \
  --set correlation-id-request-header:X-Request-Id \
  --set correlation-id-request-header:X-Correlation-Id \
  --set correlation-id-request-header:Correlation-Id \
  --set correlation-id-request-header:X-Amzn-Trace-Id
```

HTTP correlation ID example use

In this example, a request to the Directory REST API is made and the correlation ID enables finding HTTP-specific log messages with LDAP-specific log messages. The response to the API call includes a `Correlation-Id` header with the value `a54aee33-c6c6-4467-be25-efd1db7a8b76`.

```
GET /directory/v1/me?includeAttributes=mail HTTP/1.1
Accept: */*
Accept-Encoding: gzip, deflate
Authorization: Bearer ...
Connection: keep-alive
Host: localhost:1443
User-Agent: HTTPie/0.9.9

HTTP/1.1 200 OK
Content-Length: 266
Content-Type: application/hal+json
Correlation-Id: ee919049-6710-4594-9c66-28b4ada4b127
Date: Fri, 02 Nov 2018 15:16:50 GMT
Request-Id: 369
{
 "_dn": "uid=user.86,ou=People,dc=example,dc=com",
 "_links": {
  "schemas": [
   {
    "href": "https://localhost:1443/directory/v1/schemas/
    inetOrgPerson"
    }
    ],
    "self": {
    "href": "https://localhost:1443/directory/v1/
    uid=user.86,ou=People,dc=example,dc=com"
    }
    },
    "mail": [
    "user.86@example.com"
```

```
    ]
    }
```

This correlation ID can be used to search the HTTP trace log for matching log records, as shown:

```
$ grep 'correlationID="ee919049-6710-4594-9c66-28b4ada4b127"'
    PingDirectory/logs/debug-trace
    [02/Nov/2018:10:16:50.294 -0500] HTTP REQUEST requestID=369
    correlationID="ee919049-6710-4594-9c66-28b4ada4b127"
 product="Ping Identity
    Directory Server" instanceName="ds1" startupID="W9ikqA=="
 threadID=52358 from=
    [0:0:0:0:0:0:0:1]:58918 method=GET
    url="https://0:0:0:0:0:0:0:1:1443/directory/v1/me?
includeAttributes=mail"
    [02/Nov/2018:10:16:50.526 -0500] DEBUG ACCESS-TOKEN-
VALIDATOR-PROCESSING
    requestID=369
 correlationID="ee919049-6710-4594-9c66-28b4ada4b127"
    msg="Identity Mapper with DN 'cn=User ID Identity
 Mapper,cn=Identity
    Mappers,cn=config' mapped ID 'user.86' to entry DN
    'uid=user.86,ou=people,dc=example,dc=com'"
    [02/Nov/2018:10:16:50.526 -0500] DEBUG ACCESS-TOKEN-
VALIDATOR-PROCESSING
    requestID=369
 correlationID="ee919049-6710-4594-9c66-28b4ada4b127"
    accessTokenId="201811020831" msg="Token Validator 'Mock
 Access Token
    Validator' validated access token with active = 'true', sub
 = 'user.86', owner
    = 'uid=user.86,ou=people,dc=example,dc=com', clientId =
 'client1', scopes =
    'ds', expiration = 'none', not-used-before = 'none', current
 time = 'Nov 2,
    2018 10:16:50 AM CDT' "
    [02/Nov/2018:10:16:50.531 -0500] HTTP RESPONSE requestID=369
    correlationID="ee919049-6710-4594-9c66-28b4ada4b127"
    accessTokenId="201811020831" product="Ping Identity
 Directory Server"
    instanceName="ds1" startupID="W9ikqA==" threadID=52358
 statusCode=200
    etime=236.932 responseContentLength=266
    [02/Nov/2018:10:16:50.531 -0500] DEBUG HTTP-FULL-REQUEST-
AND-RESPONSE
    requestID=369
 correlationID="ee919049-6710-4594-9c66-28b4ada4b127"
    accessTokenId="201811020831" product="Ping Identity
 Directory Server"
    instanceName="ds1" startupID="W9ikqA==" threadID=52358 from=
    [0:0:0:0:0:0:0:1]:58918 method=GET

   url="https://0:0:0:0:0:0:0:1:1443/directory/v1/me?
includeAttributes=mail"
   statusCode=200 etime=236.932 responseContentLength=266 msg="
```

The LDAP log messages associated with this request can also be located:

```
$ grep 'correlationID="ee919049-6710-4594-9c66-28b4ada4b127"'
    PingDirectory/logs/access
    [02/Nov/2018:10:16:50.529 -0500] SEARCH RESULT
 instanceName="ds1"
```

```
   threadID=52358 conn=-371045 op=1657393 msgID=1657394
origin="Directory REST
   API" httpRequestID="369"
correlationID="ee919049-6710-4594-9c66-28b4ada4b127"
   authDN="uid=user.86,ou=people,dc=example,dc=com"
requesterIP="internal"
   requesterDN="uid=user.86,ou=People,dc=example,dc=com"
   requestControls="1.3.6.1.4.1.30221.2.5.2"
via="app='PingDirectoryds1'
   clientIP='0:0:0:0:0:0:0:1' sessionID='201811020831'
requestID='ee919049-6710-
   4594-9c66-28b4ada4b127'"
base="uid=user.86,ou=people,dc=example,dc=com"
   scope=0 filter="(&)" attrs="mail,objectClass" resultCode=0
   resultCodeName="Success" etime=0.684 entriesReturned=1
   [02/Nov/2018:10:16:50.530 -0500] EXTENDED RESULT
instanceName="ds1"
   threadID=52358 conn=-371046 op=1657394 msgID=1657395
origin="Directory REST
   API" httpRequestID="369"
correlationID="ee919049-6710-4594-9c66-28b4ada4b127"
   authDN="cn=Internal Client,cn=Internal,cn=Root
DNs,cn=config"
   requesterIP="internal" requesterDN="cn=Internal
Client,cn=Internal,cn=Root
   DNs,cn=config" requestControls="1.3.6.1.4.1.30221.2.5.2"
   via="app='PingDirectory-ds1' clientIP='0:0:0:0:0:0:0:1'
   sessionID='201811020831'
requestID='ee919049-6710-4594-9c66-28b4ada4b127'"
   requestOID="1.3.6.1.4.1.30221.1.6.1" requestType="Password
Policy State"
   resultCode=0 resultCodeName="Success" etime=0.542
usedPrivileges="bypassacl,
   password-reset" responseOID="1.3.6.1.4.1.30221.1.6.1"
   responseType="Password Policy State"
   dn="uid=user.86,ou=People,dc=example,dc=com"
   [02/Nov/2018:10:16:50.530 -0500] SEARCH RESULT
instanceName="ds1"
   threadID=52358 conn=-371048 op=1657397 msgID=1657398
origin="Directory REST
   API" httpRequestID="369"
correlationID="ee919049-6710-4594-9c66-28b4ada4b127"
   authDN="cn=Internal Client,cn=Internal,cn=Root
DNs,cn=config"
   requesterIP="internal" requesterDN="cn=Internal
Client,cn=Internal,cn=Root
   DNs,cn=config" requestControls="1.3.6.1.4.1.30221.2.5.2"
   via="app='PingDirectoryds1' clientIP='0:0:0:0:0:0:0:1'
   sessionID='201811020831'
requestID='ee919049-6710-4594-9c66-28b4ada4b127'"
   base="cn=Default Password Policy,cn=Password
Policies,cn=config" scope=0
   filter="(&)" attrs="dscfg- password-attribute" resultCode=0
   resultCodeName="Success" etime=0.065
preAuthZUsedPrivileges="bypassacl,
   config-read" entriesReturned=1
```

Topology configuration

Topology configuration enables grouping servers and mirroring configuration changes automatically.

Topology configuration uses a master/slave architecture for mirroring shared data across the topology. All writes and updates are forwarded to the master, which forwards them to all other servers. Reads can be served by any server in the group.

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

Topology master requirements and selection

A topology master server receives any configuration change from other servers in the topology, verifies the change, then makes the change available to all connected servers when they poll the master.

The topology master always sends a digest of its subtree contents on each update. If the node has a different digest than the master, it knows it's not synchronized. The servers will pull the entire subtree from the master if they detect that they are not synchronized. A server can detect it is not synchronized with the master under the following conditions:

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

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

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

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

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

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

Topology components

When a server is installed, it can be added to an existing topology, which will clone the server's

Topology settings are designed to operate without additional configuration. If required, you can adjust some settings to fit the needs of the environment.

Server configuration settings

Configuration settings for the topology are configured in the Global Configuration and in the Config File Handler Backend.

Though they are topology settings, they are unique to each server and are not mirrored. Settings must be the same on all servers.

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

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

`mirrored-subtree-peer-polling-interval`

Specifies the frequency at which the server polls its topology peers to determine if there are any changes that may warrant a new master selection. A lower value will ensure a faster failover, but it will also cause more traffic among the peers. The default value is five seconds. If no suitable master is found, the polling frequency is adjusted to 100 milliseconds until a new master is selected.

`mirrored-subtree-entry-update-timeout`

Specifies the maximum length of time to wait for an update operation (add, delete, modify or modify-dn) on an entry to be applied by the master on all of the servers in the topology. The default is 10 seconds. In reality, updates can take up to twice as much time as this timeout value if master selection is in progress at the time the update operation was received.

`mirrored-subtree-search-timeout`

Specifies the maximum length of time in milliseconds to wait for search operations to complete. The default is 10 seconds.

Topology settings

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

Monitor data for the topology

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

Topology data includes the following:

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

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

- The number of operations processed by this server as master.
- The number of operations processed by this server as master that were successful.
- The number of operations processed by this server as master that failed to validate.
- The number of operations processed by this server as master that failed to apply.

- The average amount of time taken (in milliseconds) by this server to process operations as the master.
- The maximum amount of time taken (in milliseconds) by this server to process an operation as the master.

Updating the server instance listener certificates
To change the SSL certificate for the server, update the keystore and truststore files with the new certificate.

About this task
The certificate file must have the new certificate in PEM-encoded format, such as:

```
-----BEGIN CERTIFICATE-----
MIIDKTCCAhGgAwIBAgIEacgGrDANBgkqhkiG9w0BAQsFADBFMR4wHAYDVQQKExVVbmJvdW5kSUQgQ2
VydGlmaWNhdGUxIzAhBgNVBAMTGnZtLW1lZGl1bS03My51bmJvdW5kaWQubGFiMB4XDTE1MTAxMjE1
MzU0OFoXDTM1MTAwNzE1MzU0OFowRTEeMBwGA1UEChMVVW5ib3VuZElEIENlcnRpZmljYXRlMSMwIQ
YDVQQDExp2bS1tZWRpdW0tNzMudW5ib3VuZGlkLmxhYjCCASIwDQYJKoZIhvcNAQEBBQADggEPADCC
AQoCggEBAKN4tAN3o9Yw6Cr9hivwVDxJqF6+aEi9Ir3WGFYLSrggRNXsiAOfWkSMWdIC5vyF5OJ9Dl
IgvHL4OuqP/YNEGzKDkgr6MwtUeVSK14+dCixygJGC0nY7k+f0WSCjtIHzrmc4WWdrZXmgb
+qv9Lup
S30JG0FXtcbGkYpjaKXIEqMg4ekz3B5cAvE0SQUFyXEdN4rWOn96nVFkb2CstbiPzAgne2tu7paJ6S
GFOW0UF7v018XY1m2WHBIoD0WC8nOVLTG9zFUavaOxtlt1TlhClkI4HRMNg8n2EtSTdQRizKuw9DdT
XJBb6Kfvnp/nI73VHRyt47wUVueehEDfLtDP8pMCAwEAAaMhMB8wHQYDVR0OBBYEFMrwjWxl2K
+yd9
+Y65oKn0g5jITgMA0GCSqGSIb3DQEBCwUAA4IBAQBpsBYodblUGew+HewqtO2i8Wt+vAbt31zM5/
kR
vo6/
+iPEASTvZdCzIBcgletxKGKeCQ0GPeHr42+erakiwmGDlUTYrU3LU5pTGTDLuR2IllTT5xlEhC
WJGWipW4q3Pl3cX/9m2ffY/
JLYDfTJaoJvnXrh7Sg719skkHjWZQgOHXlkPLx5TxFGhAovE1D4qLVR
WGohdpWDrIgFh0DVfoyAn1Ws9ICCXdRayajFI4Lc6K1m6SA5+25Y9nno8BhVPf4q5OW6+UDc8MsLbB
sxpwvR6RJ5cv3ypfOriTehJsG+9ZDo7YeqVsTVGwAlW3PiSd9bYP/8yu9Cy+0MfcWcSeAE
-----END CERTIFICATE-----
```

If clients that already have a secure connection established with this server need to be maintained, information about both certificates can reside in the same file (each with their own begin and end headers and footers). If the listener certificate needs to be updated, it might be temporarily necessary for this property to have information about the old and new certificates. This can be done by including information about both certificates in the same file, each with their own begin and end headers and footers. Blank lines, and lines that start with the # character will be ignored.

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

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

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

Steps

1. update keystore and trust store files with new SSLcert
2. Run the dsconfig to update the servers cert in the topology registry

Removing the self-signed certificate

The server is installed with a self-signed certificate and key (`ads-certificate`), which are used for internal purposes such as replication authentication, inter-server authentication in the topology registry, reversible password encryption, and encrypted backup/LDIF export.

About this task

The `ads-certificate` lives in the keystore file called `ads-truststore` under the server's `/config` directory. If your deployment requires removing the self-signed certificate, it can be replaced.

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

Steps

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

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

Prepare a new keystore with the replacement key-pair

Prepare a new keystore with an existing key-pair or using a certificate associated with the original key-pair.

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

Use an existing key-pair

If a private key and certificate in PEM-encoded format already exist, both the original private key and self-signed certificate can be replaced in `ads-truststore` with the `managecertificates` tool. The follow command imports existing certificates into a new keystore file, `ads-truststore.new`:

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

> ⓘ **Note:**

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

Use the certificate associated with the original key-pair

The certificate associated with the original sever-generated private key can be replaced with the following commands.

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

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

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

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

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

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

Updating the server configuration to use the new certificate
To update the server to use the desired key-pair, you must update the `inter-server-certificate` property for the server instance in the topology registry.

About this task

The old and the new certificates may appear within their own begin and end headers in the `inter-servercertificate` property to support transitioning from the old certificate to the new one.

Steps

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

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

2. Concatenate the old, new certificate, and issuer certificates into one file.
   - On Windows, use an editor like notepad.
   - On Unix platforms, run the command

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

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

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

Updating the ads-truststore file to use the new key-pair
The server continues to use the old `ads-certificate` until you update the certificate.

About this task

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

Steps

- To replace the old `ads-truststore` file with `ads-truststore.net`, run the command in the server's `config` directory:

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

Retiring the old certificate
Retire the old certificate when it has expired by removing it from the topology registry.

About this task

All existing encrypted backups and LDIF exports are not affected because the public key in the old and new server certificates are the same, and the private key will be able to decrypt them.

Steps

- To retire the old certificate, run the commands:

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

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

Use the configuration API

PingData servers provide a Configuration API, which can be useful in situations where using LDAP to update the server configuration is not possible.

The API is consistent with the System for Cross-domain Identity Management (SCIM) 2.0 protocol and uses `.JSON` as a text exchange format, so all request headers should allow the `application/json` content type.

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

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

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

Authentication and authorization
Clients must use HTTP Basic authentication to authenticate to the Configuration API.

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

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

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

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

Relationship between the Configuration API and the dsconfig tool
The Configuration API is designed to mirror the dsconfig tool, using the same names for properties and object types.

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

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

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

HTTP Method	Description	Related dsconfig Example
`GET`	Lists the attributes of an object when used with a path representing an object, such as `/config/globalconfiguration`or `/`	`get-backend-prop` `list-backends`

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

ⓘ **Note:**

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

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

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

GET example

Example of the GET request.

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

```
GET /config/backends/userRoot
Host: example.com:5033
```

```
Accept: application/scim+json
```

The response:

```
{
     "schemas": [
     "urn:unboundid:schemas:configuration:2.0:backend:local-db"
     ],
     "id": "userRoot",
     "meta": {
     "resourceType": "Local DB Backend",
     "location": "http://localhost:5033/config/backends/
userRoot"
     },
     "backendID": "userRoot2",
     "backgroundPrime": "false",
     "backupFilePermissions": "700",
     "baseDN": [
     "dc=example2,dc=com"
     ],
     "checkpointOnCloseCount": "2",
     "cleanerThreadWaitTime": "120000",
     "compressEntries": "false",
     "continuePrimeAfterCacheFull": "false",
     "dbBackgroundSyncInterval": "1 s",
     "dbCachePercent": "10",
     "dbCacheSize": "0 b",
     "dbCheckpointerBytesInterval": "20 mb",
     "dbCheckpointerHighPriority": "false",
     "dbCheckpointerWakeupInterval": "1 m",
     "dbCleanOnExplicitGC": "false",
     "dbCleanerMinUtilization": "75",
     "dbCompactKeyPrefixes": "true",
     "dbDirectory": "db",
     "dbDirectoryPermissions": "700",
     "dbEvictorCriticalPercentage": "0",
     "dbEvictorLruOnly": "false",
     "dbEvictorNodesPerScan": "10",
     "dbFileCacheSize": "1000",
     "dbImportCachePercent": "60",
     "dbLogFileMax": "50 mb",
     "dbLoggingFileHandlerOn": "true",
     "dbLoggingLevel": "CONFIG",
     "dbNumCleanerThreads": "0",
     "dbNumLockTables": "0",
     "dbRunCleaner": "true",
     "dbTxnNoSync": "false",
     "dbTxnWriteNoSync": "true",
     "dbUseThreadLocalHandles": "true",
     "deadlockRetryLimit": "10",
     "defaultCacheMode": "cache-keys-and-values",
     "defaultTxnMaxLockTimeout": "10 s",
     "defaultTxnMinLockTimeout": "10 s",
     "enabled": "false",
     "explodedIndexEntryThreshold": "4000",
     "exportThreadCount": "0",
     "externalTxnDefaultBackendLockBehavior": "acquire-before-
retries",
     "externalTxnDefaultMaxLockTimeout": "100 ms",
     "externalTxnDefaultMinLockTimeout": "100 ms",
     "externalTxnDefaultRetryAttempts": "2",
     "hashEntries": "false",
     "id2childrenIndexEntryLimit": "66",
```

```
     "importTempDirectory": "import-tmp",
     "importThreadCount": "16",
     "indexEntryLimit": "4000",
     "isPrivateBackend": "false",
     "javaClass":
 "com.unboundid.directory.server.backends.jeb.BackendImpl",
     "jeProperty": [
     "je.cleaner.adjustUtilization=false",
     "je.nodeMaxEntries=32"
     ],
     "numRecentChanges": "50000",
     "offlineProcessDatabaseOpenTimeout": "1 h",
     "primeAllIndexes": "true",
     "primeMethod": [
     "none"
     ],
     "primeThreadCount": "2",
     "primeTimeLimit": "0 ms",
     "processFiltersWithUndefinedAttributeTypes": "false",
     "returnUnavailableForUntrustedIndex": "true",
     "returnUnavailableWhenDisabled": "true",
     "setDegradedAlertForUntrustedIndex": "true",
     "setDegradedAlertWhenDisabled": "true",
     "subtreeDeleteBatchSize": "5000",
     "subtreeDeleteSizeLimit": "5000",
     "uncachedId2entryCacheMode": "cache-keys-only",
     "writabilityMode": "enabled"
     }
```

GET list example

Example of the GET request for all local backends.

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

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

The response, which has been shortened:

```
{
    "schemas": [
    "urn:ietf:params:scim:api:messages:2.0:ListResponse"
    ],
    "totalResults": 24,
    "Resources": [
    {
    "schemas": [
    "urn:unboundid:schemas:configuration:2.0:backend:ldif"
    ],
    "id": "adminRoot",
    "meta": {
    "resourceType": "LDIF Backend",
    "location": "http://localhost:5033/config/backends/
adminRoot"
    },
    "backendID": "adminRoot",
    "backupFilePermissions": "700",
    "baseDN": [
    "cn=admin data"
    ],
```

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

PATCH example

Configuration can be modified using the HTTP PATCH method.

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

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

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

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

```
}
```

is analogous to:

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

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

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

is analogous to:

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

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

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

is analogous to:

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

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

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

is analogous to:

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

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

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

is analogous to:

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

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

Example request:

```
PATCH /config/backends/userRoot
    Host: example.com:5033
    Accept: application/scim+json
    {
    "schemas" :
 [ "urn:ietf:params:scim:api:messages:2.0:PatchOp" ],
    "Operations" : [ {
    "op" : "replace",
    "path" : "description",
    "value" : "A new backend."
    }, {
    "op" : "add",
    "path" : "jeProperty",
    "value" : "je.env.backgroundReadLimit=0"
    }, {
    "op" : "remove",
    "path" : "[jeProperty eq
 \"je.cleaner.adjustUtilization=false\"]"
    }, {
    "op" : "remove",
    "path" : "jeProperty[value eq \"je.nodeMaxEntries=32\"]"
    }, {
    "op" : "remove",
    "path" : "id2childrenIndexEntryLimit"
    } ]
    }
```

Example response:

```
{
    "schemas": [
    "urn:unboundid:schemas:configuration:2.0:backend:local-db"
    ],
    "id": "userRoot2",
    "meta": {
    "resourceType": "Local DB Backend",
    "location": "http://example.com:5033/config/backends/
userRoot2"
    },
    "backendID": "userRoot2",
    "backgroundPrime": "false",
    "backupFilePermissions": "700",
    "baseDN": [
    "dc=example2,dc=com"
    ],
    "checkpointOnCloseCount": "2",
    "cleanerThreadWaitTime": "120000",
    "compressEntries": "false",
    "continuePrimeAfterCacheFull": "false",
    "dbBackgroundSyncInterval": "1 s",
    "dbCachePercent": "10",
    "dbCacheSize": "0 b",
    "dbCheckpointerBytesInterval": "20 mb",
    "dbCheckpointerHighPriority": "false",
    "dbCheckpointerWakeupInterval": "1 m",
    "dbCleanOnExplicitGC": "false",
```

```
    "dbCleanerMinUtilization": "75",
    "dbCompactKeyPrefixes": "true",
    "dbDirectory": "db",
    "dbDirectoryPermissions": "700",
    "dbEvictorCriticalPercentage": "0",
    "dbEvictorLruOnly": "false",
    "dbEvictorNodesPerScan": "10",
    "dbFileCacheSize": "1000",
    "dbImportCachePercent": "60",
    "dbLogFileMax": "50 mb",
    "dbLoggingFileHandlerOn": "true",
    "dbLoggingLevel": "CONFIG",
    "dbNumCleanerThreads": "0",
    "dbNumLockTables": "0",
    "dbRunCleaner": "true",
    "dbTxnNoSync": "false",
    "dbTxnWriteNoSync": "true",
    "dbUseThreadLocalHandles": "true",
    "deadlockRetryLimit": "10",
    "defaultCacheMode": "cache-keys-and-values",
    "defaultTxnMaxLockTimeout": "10 s",
    "defaultTxnMinLockTimeout": "10 s",
    "description": "123",
    "enabled": "false",
    "explodedIndexEntryThreshold": "4000",
    "exportThreadCount": "0",
    "externalTxnDefaultBackendLockBehavior": "acquire-before-
retries",
    "externalTxnDefaultMaxLockTimeout": "100 ms",
    "externalTxnDefaultMinLockTimeout": "100 ms",
    "externalTxnDefaultRetryAttempts": "2",
    "hashEntries": "false",
    "importTempDirectory": "import-tmp",
    "importThreadCount": "16",
    "indexEntryLimit": "4000",
    "isPrivateBackend": "false",
    "javaClass":
 "com.unboundid.directory.server.backends.jeb.BackendImpl",
    "jeProperty": [
    "\"je.env.backgroundReadLimit=0\""
    ],
    "numRecentChanges": "50000",
    "offlineProcessDatabaseOpenTimeout": "1 h",
    "primeAllIndexes": "true",
    "primeMethod": [
    "none"
    ],
    "primeThreadCount": "2",
    "primeTimeLimit": "0 ms",
    "processFiltersWithUndefinedAttributeTypes": "false",
    "returnUnavailableForUntrustedIndex": "true",
    "returnUnavailableWhenDisabled": "true",
    "setDegradedAlertForUntrustedIndex": "true",
    "setDegradedAlertWhenDisabled": "true",
    "subtreeDeleteBatchSize": "5000",
    "subtreeDeleteSizeLimit": "5000",
    "uncachedId2entryCacheMode": "cache-keys-only",
    "writabilityMode": "enabled",
    "urn:unboundid:schemas:configuration:messages:2.0": {
    "requiredActions": [
    {
    "property": "jeProperty",
    "type": "componentRestart",
    "synopsis": "In order for this modification to take effect,
```

```
    the component must be restarted, either by disabling and
    re-enabling it, or by restarting the server"
    },
    {
    "property": "id2childrenIndexEntryLimit",
    "type": "other",
    "synopsis": "If this limit is increased, then the contents
    of the backend must be exported to LDIF and re-imported to
    allow the new limit to be used for any id2children keys
    that had already hit the previous limit."
   }
  ]
 }
}
```

API paths

The Configuration API is available under the /config path.

A full listing of root sub-paths can be obtained from the `/config/ResourceTypes` endpoint:

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

Sample response (abbreviated):

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

```
    "endpoint": "/access-control-handler",
    "meta": {
    "resourceType": "ResourceType",
    "location": "http://example.com:5033/config/ResourceTypes/accesscontrol-
    handler"
    }
    },
    {
    ...
```

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

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

Sort and filter configuration objects
The Configuration API supports System for Cross-domain Identity Management (SCIM) parameters for filter, sorting, and pagination.

Search operations can specify a SCIM filter used to narrow the number of elements returned. See the SCIM specification for the full set of operations for SCIM filters. Clients may also specify sort parameters, or paging parameters. As previously mentioned, clients may specify attributes to include or exclude in both get and list operations.

GET parameters for sorting and filtering

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

Update properties
The Configuration API supports the HTTP PUT method as an alternative to modifying objects with HTTP PATCH.

With PUT, the server computes the differences between the object in the request with the current version in the server, and performs modifications where necessary. The server will never remove attributes that are not specified in the request. The API responds with the entire modified object.

Request:

```
PUT /config/backends/userRoot
Host: example.com:5033
```

```
Accept: application/scim+json
{
  "description" : "A new description."
  }
```

Response:

```
{
   "schemas": [
   "urn:unboundid:schemas:configuration:2.0:backend:local-db"
   ],
   "id": "userRoot",
   "meta": {
   "resourceType": "Local DB Backend",
   "location": "http://example.com:5033/config/backends/
userRoot"
   },
   "backendID": "userRoot",
   "backgroundPrime": "false",
   "backupFilePermissions": "700",
   "baseDN": [
   "dc=example,dc=com"
   ],
   "checkpointOnCloseCount": "2",
   "cleanerThreadWaitTime": "120000",
   "compressEntries": "false",
   "continuePrimeAfterCacheFull": "false",
   "dbBackgroundSyncInterval": "1 s",
   "dbCachePercent": "25",
   "dbCacheSize": "0 b",
   "dbCheckpointerBytesInterval": "20 mb",
   "dbCheckpointerHighPriority": "false",
   "dbCheckpointerWakeupInterval": "30 s",
   "dbCleanOnExplicitGC": "false",
   "dbCleanerMinUtilization": "75",
   "dbCompactKeyPrefixes": "true",
   "dbDirectory": "db",
   "dbDirectoryPermissions": "700",
   "dbEvictorCriticalPercentage": "5",
   "dbEvictorLruOnly": "false",
   "dbEvictorNodesPerScan": "10",
   "dbFileCacheSize": "1000",
   "dbImportCachePercent": "60",
   "dbLogFileMax": "50 mb",
   "dbLoggingFileHandlerOn": "true",
   "dbLoggingLevel": "CONFIG",
   "dbNumCleanerThreads": "1",
   "dbNumLockTables": "0",
   "dbRunCleaner": "true",
   "dbTxnNoSync": "false",
   "dbTxnWriteNoSync": "true",
   "dbUseThreadLocalHandles": "true",
   "deadlockRetryLimit": "10",
   "defaultCacheMode": "cache-keys-and-values",
   "defaultTxnMaxLockTimeout": "10 s",
   "defaultTxnMinLockTimeout": "10 s",
   "description": "abc",
   "enabled": "true",
   "explodedIndexEntryThreshold": "4000",
   "exportThreadCount": "0",
   "externalTxnDefaultBackendLockBehavior": "acquire-before-
retries",
   "externalTxnDefaultMaxLockTimeout": "100 ms",
```

```
    "externalTxnDefaultMinLockTimeout": "100 ms",
    "externalTxnDefaultRetryAttempts": "2",
    "hashEntries": "true",
    "importTempDirectory": "import-tmp",
    "importThreadCount": "16",
    "indexEntryLimit": "4000",
    "isPrivateBackend": "false",
    "javaClass":
  "com.unboundid.directory.server.backends.jeb.BackendImpl",
    "numRecentChanges": "50000",
    "offlineProcessDatabaseOpenTimeout": "1 h",
    "primeAllIndexes": "true",
    "primeMethod": [
    "none"
    ],
    "primeThreadCount": "2",
    "primeTimeLimit": "0 ms",
    "processFiltersWithUndefinedAttributeTypes": "false",
    "returnUnavailableForUntrustedIndex": "true",
    "returnUnavailableWhenDisabled": "true",
    "setDegradedAlertForUntrustedIndex": "true",
    "setDegradedAlertWhenDisabled": "true",
    "subtreeDeleteBatchSize": "5000",
    "subtreeDeleteSizeLimit": "100000",
    "uncachedId2entryCacheMode": "cache-keys-only",
    "writabilityMode": "enabled"
    }
```

Administrative actions

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

Changing the `jeProperty` of a backend will result in the following:

```
"urn:unboundid:schemas:configuration:messages:2.0": {
  "requiredActions": [
    {
      "property": "baseContextPath",
      "type": "componentRestart",
      "synopsis": "In order for this modification to
  take effect, the component must be restarted,
  either by disabling and re-enabling it, or by
  restarting the server"
    },
    {
 "property": "id2childrenIndexEntryLimit",
 "type": "other",
 "synopsis": "If this limit is increased, then the
  contents of the backend must be exported to LDIF
  and re-imported to allow the new limit to be used
  for any id2children keys that had already hit the
  previous limit."
    }
  ]
}
...
```

Update servers and server groups

Servers can be configured as part of a server group, so that configuration changes that are applied to a single server, are then applied to all servers in a group.

When managing a server that is a member of a server group, creating or updating objects using the Configuration API requires the `applyChangeTo` query parameter. The behavior and acceptable values for this parameter are identical to the `dsconfig` parameter of the same name. A value of `singleServer` or `serverGroup` can be specified. For example:

```
https://example.com:5033/config/Backends/userRoot?applyChangeTo=singleServer
```

> ⓘ **Note:**
>
> This does not apply to mirrored subtree objects, which include Topology and Cluster level objects. Changes made to mirrored objects are applied to all objects in the subtree.

Configuration API responses

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

Response code	Description	Response body
200 Success	The requested operation succeeded, with the response body being the configuration object that was created or modified. If further actions are required, they are included in the `urn:unboundid:schemas:configuration:messages:2.0` object.	List of objects, or object properties, administrative actions.
204 No Content	The requested operation succeeded and no further information has been provided, such as in the case of a DELETE operation.	None.
400 Bad Request	The request contents are incorrectly formatted or a request is made for an invalid API version.	Error summary and optional message.
401 Unauthorized	User authentication is required. Some user agents such as browsers can respond by prompting for credentials. If the request had specified credentials in an Authorization header, they are invalid.	None.
403 Forbidden	The requested operation is forbidden either because the user does not have sufficient privileges or some other constraint such as an object is edit-only and cannot be deleted.	None.
404 Not Found	The requested path does not refer to an existing object or object relation.	Error summary and optional message.

Response code	Description	Response body
409 Conflict	The requested operation could not be performed due to the current state of the configuration. For example, an attempt was made to create an object that already exists or an attempt was made to delete an object that is referred to by another object.	Error summary and optional message.
415 Unsupported Media Type	The request is such that the Accept header does not indicate that JSON is an acceptable format for a response.	None.
500 Server Error	The server encountered an unexpected error. Please report server errors to customer support.	Error summary and optional message.

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

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

Domain name service (DNS) caching

If needed, you can use two global configuration properties to control the caching of hostname-to-numeric IP address (DNS lookup) results returned from the name resolution services of the underlying operating system.

Use the `dsconfig` tool to configure these properties:

`network-address-cache-ttl`

Sets the Java system property `networkaddress.cache.ttl`, and controls the length of time in seconds that a hostname-to-IP address mapping can be cached. The default behavior is to keep resolution results for one hour (3600 seconds). This setting applies to the server and all extensions loaded by the server.

`network-address-outage-cache-enabled`

Caches hostname-to-IP address results in the event of a DNS outage. This is set to `true` by default, meaning name resolution results are cached. Unexpected service interruptions can occur during planned or unplanned maintenance, network outages or an infrastructure attack. This cache may allow the server to function during a DNS outage with minimal impact. This cache is not available to server extensions.

IP address reverse name lookups

Address masks configured in Access Control Lists (ACIs), Connection Handlers, Connection Criteria, and Certificate handshake processing can trigger implicit reverse name lookups.

Ping Identity servers do not explicitly perform numeric IP address-to-hostname lookups. For more information about how address masks are configured in the server, review the following information for each server:

- ACI dns: Bind rules under Managing Access Control (PingDirectory Server and PingDirectoryProxy Servers)
- `ds-auth-allowed-address`: Adding Operational Attributes that Restrict Authentication (PingDirectory Server)
- Connection criteria: Restricting Server Access Based on Client IP Address (PingDirectory Server and PingDirectoryProxy Servers)
- Connection handlers: Restrict server access using Connection Handlers (Configuration Reference Guide for all PingData servers)

Configure traffic through a load balancer

If a PingData server is sitting behind an intermediate HTTP server, such as a load balancer, a reverse proxy, or a cache, it will log incoming requests as originating with the intermediate HTTP server instead of the client that actually sent the request.

If the actual client's IP address should be recorded to the trace log, enable `X-Forwarded-*` handling in both the intermediate HTTP server and the PingData server. See the product documentation for the device type. For PingData servers:

- Edit the appropriate Connection Handler object (HTTPS or HTTP) and set `use-forwarded-headers` to `true`.
- When `use-forwarded-headers` is set to `true`, the server will use the client IP address and port information in the `X-Forwarded-*` headers instead of the address and port of the entity that's actually sending the request, the load balancer. This client address information will show up in logs where one would normally expect it to show up, such as in the `from` field of the HTTP REQUEST and HTTP RESPONSE messages.

Configuring authentication with a SASL external certificate

Configure the PingDataMetrics Server to use SASL EXTERNAL to authenticate to the PingDirectory Server with a client certificate.

About this task

By default, the PingDataMetrics Server authenticates to the PingDirectory Server using LDAP simple authentication (with a bind DN and a password).

> ⓘ **Note:**
>
> This procedure assumes that PingDataMetrics Server instances are installed and configured to communicate with the backend PingDirectory Server instances using either SSL or StartTLS.

After the servers are configured, perform the following steps to configure SASL EXTERNAL authentication:

Steps

1. Create a Java KeyStore (JKS) that includes a public and private key pair for a certificate that the PingDataMetrics Server instance will use to authenticate to the PingDirectory Server instance.
   a. Run the following command in the instance root of one of the PingDataMetrics Server instances.

```
$ keytool -genkeypair \
```

```
  -keystore config/metrics-user-keystore \
  -storetype JKS \
  -keyalg RSA \
  -keysize 2048 \
  -alias metrics-user-cert \
  -dname "cn=Metrics User,cn=Root DNs,cn=config" \
  -validity 7300
```

   b. When prompted for a keystore password, enter a strong password to protect the certificate.
   c. When prompted for the key password, press `Enter` to use the keystore password to protect the private key.
2. Create a `config/metrics-user-keystore.pin` file that contains a single line that is the keystore password provided in the previous step.
3. If there are other PingDataMetrics Server instances in the topology, copy the `metrics-user-keystore` and `metrics-user-keystore.pin` files into the config directory for all instances.
4. Run the following command to export the public component of the user certificate to a text file:

```
$ keytool -export \
  -keystore config/metrics-user-keystore \
  -alias metrics-user-cert \
  -file config/metrics-user-cert.txt
```

5. Copy the `metrics-user-cert.txt` file into the config directory of all PingDirectory Server instances.
   a. Import that certificate into each server's primary trust store by running the following command from the server root.

```
$ keytool -import \
  -keystore config/truststore \
  -alias metrics-user-cert \
  -file config/metrics-user-cert.txt
```

   b. When prompted for the keystore password, enter the password contained in the `config/truststore.pin` file.
   c. When prompted to trust the certificate, enter `yes`.
6. Update the configuration for each PingDataMetrics Server instance to create a new key manager provider that will obtain its certificate from the `config/metrics-user-keystore` file.
   a. Run the following `dsconfig` command from the server root:

```
$ dsconfig create-key-manager-provider \
  --provider-name "Metrics User Certificate" \
  --type file-based \
  --set enabled:true \
  --set key-store-file:config/metrics-user-keystore \
  --set key-store-type:JKS \
  --set key-store-pin-file:config/metrics-user-keystore.pin
```

7. Update the configuration for each LDAP external server in each PingDataMetrics Server instance to use the newly-created key manager provider, and also to use SASL EXTERNAL authentication instead of LDAP simple authentication.
   a. Run the following `dsconfig` command:

```
$ dsconfig set-external-server-prop \
  --server-name ds1.example.com:636 \
  --set authentication-method:external \
  --set "key-manager-provider:Metrics User Certificate"
```

   After these changes, the PingDataMetrics Server should re-establish connections to the LDAP external server and authenticate with SASL EXTERNAL.

8. Verify that the PingDataMetrics Server is still able to communicate with all backend servers by running the `bin/status` command.

   ⓘ **Note:**

   All of the servers listed in the "--- LDAP External Servers ---" section should have a status of `Available`. Review the PingDirectory Server access log can to make sure that the BIND RESULT log messages used to authenticate the connections from the PingDataMetrics Server include `authType="SASL", saslMechanism="EXTERNAL", resultCode=0`, and `authDN="cn=Metrics User,cn=Root DNs,cn=config"`.

Server SDK extensions

Custom server extensions can be created with the Server SDK.

Extension bundles are installed from a `.zip` archive or a file system directory. Use the `manage-extension` tool to install or update any extension that is packaged using the extension bundle format. It opens and loads the extension bundle, confirms the correct extension to install, stops the server if necessary, copies the bundle to the server install root, and then restarts the server.

ⓘ **Note:**

The `manage-extension` tool must be used with Java extensions packaged using the extension bundle format. For more information, see the "Building and Deploying Java-Based Extensions" section of the Server SDK documentation.

The Server SDK enables creating extensions for all PingData servers. Cross-product extensions include:

- Access loggers
- Alert handlers
- Error loggers
- Key manager providers
- Monitor providers
- Trust manager providers
- OAuth token handlers
- Manage extension plugins

Collecting data and metrics

The PingDataMetrics Server polls all the monitored servers over LDAP to gather the following data:

- Status of each server
- Alerts emitted by each server
- Performance data exposed in the `cn=monitor` subtree of each server

For a complete summary of the metrics and dimensions that can be exposed through the RESTful Metrics API, see the reference files located in the `docs/metrics-guide` directory. Most metrics have a count, minimum, maximum, and average.

Metrics overview

A metric corresponds to a single measurement made within the server.

The PingDataMetrics Server collects three types of metrics:

Count metrics

Represent the number of times a specific event happens within the server. Examples of count metrics include the number of LDAP operations performed, network packets received, or new connections established.

Continuous-valued metrics

Measure things that always have a value. For example, these metrics include the amount of free disk space, the current number of connected clients, and the number of operations pending in the work queue.

Discrete metrics

Correspond to measurements that have both a value and a weight, such as the duration of an LDAP operation or the average duration of a checkpoint.

The statistics that can be applied to values depend on the metric type. Only count statistics are available for count metrics. Discrete metrics have count, average, and histogram statistics available, which expose a count of the values broken down into bucket ranges. Average, minimum, and maximum statistics are available for continuous-valued metrics.

Count metrics
A count metric indicates the number of times a specific event happens within the server.

For example, the number of packets received on a network interface during a measurement interval is a count metric. Each measurement returns the count of the number of packets received during that measurement interval only. The sample contains the number of occurrences, whether the measurement interval is five seconds or two minutes.

Another example of a count metric is the number of megabytes of data written to a disk device during a measurement interval. Using the COUNT statistic when querying for a count metric will return the sum of the counts. Count metrics can often be converted into a rate.

Continuous metrics
A continuous metric is a measurement of a value where the thing being measured always has a valid value at each measurement point.

For example, CPU percent busy is a continuous metric. For every sample CPU interval, a valid CPU percent busy measurement can be taken. A continuous metric differs from a count metric in that continuous metric samples cannot be added across time in a meaningful way. Instead, continuous metric samples use average, minimum, and maximum statistics. To determine how busy the CPU has been since midnight, average, rather than sum, the samples since midnight.

Discrete metrics
A discrete metric is a measurement that has both a value and a weight.

Discrete metrics are different from continuous metrics because each measurement is weighted. A discrete metric is analogous to a weighted average and requires that multiple measurements be taken within a single sample interval. For example, LDAP operation response time is a discrete metric, where the actual response time of each operation is averaged, and the number of LDAP operations is provided as the weight. If no LDAP operations occur in a sample interval, the value would be zero and the weight would be zero.

Some continuous and discrete metrics may also report a minimum/maximum value if the measurement is composed of multiple sub-measurements. The minimum/maximum values are aggregated by averaging, so the values reflect the median.

Some discrete metrics may also convey histogram data. Histogram data represents an additional set of measurements that take individual measurements and place them into value ranges. The PingDataMetrics Server supports histograms with up to 15 value ranges. Histogram valued samples are unique because they give a picture of the distribution of the values, and because they more precisely answer the question of "How many samples are greater than X?"

Dimensions

Dimensions provide a means of aggregating and subdividing metric sample values in a way that logically follows what is actually measured.

For example, metrics that measure disk activity have a disk-device dimension. Aggregating on the disk-device dimension shows the average disk activity for all disks, where pivoting (splitting) by the disk-device dimension shows the activity for specific disks.

Every metric has a logical instance dimension, which corresponds to the server on which the sample was created. Each metric may have up to three dimensions, which are defined in the metric definition.

For example, the `sync-pipe-completed-ops` metric has two dimensions, the `pipe-name` and `pipe-result`. The `pipe-name` is the name of the sync pipe as configured for the PingDataSync Server. The `pipe-result` is one of the following values:

- `exception`
- `failed`
- `failed-at-resource`
- `failed-during-mapping`
- `match-multiple-at-dest`
- `no-match-at-dest`
- `already-exists-at-dest`
- `no-change-needed`
- `out-of-scope`
- `success`
- `aborted-by-plugin`
- `failed-in-plugin`

At each measurement interval for each sync pipe on each PingDataSync Server, there will be a value for each of the `pipe-result` values. So, for a single PingDataSync Server with two Sync Pipes, `pipe-one` and `pipe-two`, the samples generated for each sample period look like the following. The timestamp is constrained to time-only for brevity.

```
08:15:05, sync-pipe-completed-ops, pipe-one, exception, 1
08:15:05, sync-pipe-completed-ops, pipe-one, failed, 7
08:15:05, sync-pipe-completed-ops, pipe-one, failed-at-resource, 1
08:15:05, sync-pipe-completed-ops, pipe-one, failed-during-mapping, 1
08:15:05, sync-pipe-completed-ops, pipe-one, match-multiple-at-dest, 3
08:15:05, sync-pipe-completed-ops, pipe-one, no-match-at-dest, 0
08:15:05, sync-pipe-completed-ops, pipe-one, already-exists-at-dest, 0
08:15:05, sync-pipe-completed-ops, pipe-one, no-change-needed, 1
08:15:05, sync-pipe-completed-ops, pipe-one, out-of-scope, 1
08:15:05, sync-pipe-completed-ops, pipe-one, success, 125
08:15:05, sync-pipe-completed-ops, pipe-one, aborted-by-plugin, 1
08:15:05, sync-pipe-completed-ops, pipe-one, failed-in-plugin, 0
08:15:05, sync-pipe-completed-ops, pipe-two, exception, 3
08:15:05, sync-pipe-completed-ops, pipe-two, failed, 9
08:15:05, sync-pipe-completed-ops, pipe-two, failed-at-resource, 2
08:15:05, sync-pipe-completed-ops, pipe-two, failed-during-mapping, 1
08:15:05, sync-pipe-completed-ops, pipe-two, match-multiple-at-dest, 4
08:15:05, sync-pipe-completed-ops, pipe-two, no-match-at-dest, 0
08:15:05, sync-pipe-completed-ops, pipe-two, already-exists-at-dest, 0
08:15:05, sync-pipe-completed-ops, pipe-two, no-change-needed, 1
08:15:05, sync-pipe-completed-ops, pipe-two, out-of-scope, 1
08:15:05, sync-pipe-completed-ops, pipe-two, success, 217
08:15:05, sync-pipe-completed-ops, pipe-two, aborted-by-plugin, 1
08:15:05, sync-pipe-completed-ops, pipe-two, failed-in-plugin, 0
```

Compare how busy `pipe-one` is to `pipe-two` by pivoting on `pipe-name`. This results in the following:

```
pipe-one 141
pipe-two 239
```

Pivot by `pipe-result`, to get a set of counts that show the distribution of the counts of the specific error types, as well as the success and failure. This data provides a quick way of assessing the kinds of problems encountered by the Sync Pipes.

Dimensions provide a way to pivot or aggregate along a metric-specific axis. All metrics have the `instance` pivot and the `time` pivot. Metrics that support the histogram statistic can also have a `histogram` pivot.

Query overview

Overview of the three components of a metric query.

A metric query consists of three components:

- The data used to calculate the query results
- The aggregation method used on the data to calculate the query result
- The format of the query result

Select query data

The data used to generate the results of a metric query are driven by the following factors:

- Metric and statistic
- Time range
- Server instances included in the result
- Included dimension values
- Histogram range

Every query returns results for a single statistic and of a single metric. A query must include the time range used to generate the results. Time ranges can either be absolute dates (in ISO-8601 format) or relative dates (such as -30m). A relative start time offset is relative to the end time. A relative end time offset is relative to the current time. When no end time is specified, the server includes results up to the current time.

The time range and the desired number of points (for pivot by time) dictates the resolution of data used to process the query. For example, the finest granularity of data, one second resolution, is only kept for a few hours. It will not be used to satisfy a query spanning multiple days.

By default, all server instances that produce the metric are used to calculate the query results. However, the metric query can be restricted to one of the following:

- A specific list of servers
- Servers of a given type, such as PingDirectory server
- Servers within a specific location

For metrics that include one or more dimensions, a query can be evaluated across a subset of dimension values. For example, the results returned for the `response-time` metric can be restricted to just the search and modify values of the `op-type` dimension.

For `discrete-valued` metrics that break their values down into histogram ranges, a query can count statistics applied to a subset of histogram buckets by specifying a minimum and/or maximum histogram value. For example, a query on the `response-time` metric could return a count of operations that took longer than 100 milliseconds.

Aggregate query results
A metric query can return the full, raw data that matches the query parameters, so that the server can aggregate metric results across time, server instance, dimension value, or histogram value.

The server aggregates results, except when the query indicates not to, by using a pivot. The mechanism for aggregating the data depends on the type of metric. A pivot directs the query processor to not aggregate one component of the query data. A pivot can be based on time, server instance, a specific dimension, or histogram ranges.

- If no pivot is specified, the query returns a single number that represents the aggregation of all matched data. For example, a query with no pivot might return the total number of operations that have completed today.
- A single pivot results in one-dimensional data, such as a time-based chart with a single line or a simple bar chart.
- Two pivots results in two-dimensional data, such as a time-based chart with a separate line for each server instance, or a stacked bar chart that shows the number of completed operations broken down by server and operation type.
- Three pivots results in three-dimensional data, such as a stacked, grouped bar chart that shows completed operations broken down by server, operation type, and result.

Beyond aggregating multiple samples into one, the data returned by a metric query can be further manipulated.

> You can scale queries on the count statistic to return the count of events per second, per minute, or per hour. Counts of histogram values can be returned by a percentage of the total. For instance, instead of returning the raw count of operations that took longer than 50 milliseconds to complete, the results could return as the percentage of all operations that took longer than 50 milliseconds to complete. A value of 0.02% is more meaningful than a value of 40

Format query results
The query results can be converted into a format requested by the client.

Query result formats include:

- CSV spreadsheet
- XML format
- JSON format

The query-metric tool

The `query-metric` tool is a client application of the PingDataMetrics server API that enables access to all the metrics gathered by the server.

The `query-metric` tool includes subcommands that facilitate creating data queries for listing metrics, server instances, and dimension values. This tool runs in both interactive and non-interactive modes. Queries are formed using the following subcommands:

`explore`

Creates a series of hyper-linked HTML files for a broad range of metrics. The tool generates these files by making a series of API queries for a set of servers and metrics. The tool highlights the breadth of available metrics and patterns or anomalies across multiple metrics. In interactive mode, the tool prompts for the servers and the metrics.

`query`

Defines a query for specific data of interest. In interactive mode, the tool prompts for the server, metrics, dimensions, statistics, and pivot values. The tool can be used to request a data formatted in `.xml`, `.json`, or `.csv`.

`query-metric` tool commands

To start the tool in interactive mode, enter the following command:

```
$ query-metric
```

Or, specify a subcommand in interactive mode:

```
$ query-metric explore
```

In non-interactive mode, the tool generates a data table based on command-line input. For example, the following command requests information from the local PingDataMetrics server listening on port 8080 and generates response-time and throughput data tables for Proxy server instances in Austin for the previous two weeks:

```
$ query-metric explore \
  --httpPort 8080 \
  --instanceType proxy \
  --instanceLocation Austin \
  --metric response-time \
  --metric throughput \
  --startTime -2w
```

The following command line obtains a JSON formatted data table that shows average throughput for all PingDirectoryProxy server instances, over time with 100 data points. Each line in the table represents either an application's search or modification throughput. Throughput values are represented as operations per second:

```
$ query-metric query \
  --hostname localhost \
  --httpPort 8080 \
  --username cn=user1,cn=api-users \
  --password secret \
  --table json \
  --metric throughput \
  --instanceType proxy \
  --statistic average \
  --pivot op-type \
  --pivot application-name \
  --dimension op-type:search,modify \
  --rateScaling second \
  --maxIntervals 100 \
  --startTime 2012-09-01T17:41Z \
  --endTime 2012-09-30T17:41Z
```

To see a list of all supported options, run the help option for the query-metric tool:

```
$ query-metric -?
```

Performance data collection

Performance data represents a majority of the data collected by the PingDataMetrics server.

Each server can produce hundreds of kilobytes of performance data per minute, though the amount of data captured has little to no impact on the performance of the monitored system. By default, the PingDataMetrics server stores performance data for 20 years. Configure the volume of performance data collected by each monitored server so that the PingDataMetrics server can keep up with the flow.

The performance data model is a dimensional data model. Measurements can be taken on multiple simultaneous values that are distinguished by dimension values. For example, a response time metric

provides the time in milliseconds it took a server to respond to an LDAP request. This response-time metric has two dimensions:

Application name

Reflects the connection criteria of the request.

Operation type

Corresponds to the LDAP operation, such as add, bind, or search. If a server has 20 different connection criteria, each response-time sample may have 140 different values, one for each of the applications multiplied by the number of operation types.

The performance data captured on the monitored server has a record with the following fields.

Performance data fields

Name	Data type	Description
Timestamp	Date	Time of measurement, using clock on the monitored server
Metric	String	Name of metric
Dimension	String	Values of dimensions 1 - 3
Count	Int	Number of measurements represented by this sample
Average	Double	Average value of this sample
Minimum	Double	Optional minimum value of this sample
Maximum	Double	Optional maximum value of this sample
Buckets	Int	Optional histogram data associated with this sample

When a performance record is imported into the PingDataMetrics server, it is normalized to reduce the size of the record. The normalized record contains the following information.

Normalized record in the PingDataMetrics Server

Name	Data type	Description
`batchID`	Int	The ID of the batch of data to which this record belongs
`sampleTime`	Timestamp	The time the sample was captured or equivalent information after aggregation
`metric_qual`	Int	The ID of a structure that reflects the metric and all dimension values
`definitionID`	Int	ID of the histogram definition, if the data belong to a histogram-valued sample

Name	Data type	Description
`count`	Int	Number of measurements represented by this sample
`avg_val`	Real	Average value for this sample
`min_val`	Real	Minimum value for this sample
`max_val`	Real	Maximum value for this sample
`val 1-15`	Long	Histogram bucket values

System monitoring data collection

All servers have the ability to monitor their health and that of the host system.

Servers do not collect any performance data until they are prepared by the PingDataMetrics server. All of the important server and machine metrics are stored in the `cn=monitor` backend.

Stats Collector plugin

The Stats Collector plugin is the primary driver of performance data collection for LDAP, server response, replication, local JE databases, and host system machine metrics.

Stats Collector configuration determines the sample and collection intervals, granularity of data (basic, extended, verbose), types of host system collection (cpu, disk, network) and the type of data aggregation that occurs for LDAP application statistics. The Stats Collector plugin is configured with the `dsconfig` tool and collects data using LDAP queries. For example, the `--server-info:extended` option includes collection for the following:

- CPU
- Java virtual machine (JVM) memory
- Memory
- Disk information
- Network information

The following are all options for the Stats Collector plugin:

```
>>>> Configure the properties of the Stats Collector Plugin
Property             Value(s)
-------------------------------------------------------------------------
1) description           -
2) enabled             false
3) local-db-backend-info        basic
4) replication-info        basic
5) entry-cache-info        basic
6) host-info        cpu, disk, network
7) included-ldap-application        If per-application LDAP stats is
 enabled,
                                       then stats will be included for all
 applications.
8) sample-interval        1 s
9) collection-interval        500 ms
10) ldap-info        extended
11) server-info        basic
12) per-application-ldap-stats        aggregate-only
```

System utilization monitors
The System Utilization Monitors interface directly with the host operating system to gather statistics about CPU utilization and idle states, memory consumption, disk input and output rates, and queue depths, as well as network packet transmit and receive activity.

Utilization metrics are gathered with externally invoked operating system commands using platform-specific arguments and version-specific output parsing, such as `iostat` and `netstat`.

Enabling the Host System monitor provider automatically gathers CPU and memory utilization, but only optionally gathers disk and network information. Disk and network interfaces are enumerated in the configuration by device names (such as `eth0` or `lo`), and by disk device names (such as `sd1, sdab, sda2, scsi0`).

External collector daemon
The System Utilization monitor contains an embedded collector daemon that runs on systems affected by a Java process fork memory issue (RFE 5049299).

When a process attempts to fork a child process, the system attempts to allocate the same amount of memory for the child process, which will likely fail when the parent process consumes a large amount of memory.

The embedded collector daemon is started automatically for the server and inspects the Host System Monitor provider configuration to conditionally determine whether the external daemon process is required.

The external collector daemon operates by having an internal table of repeatable commands that run on a schedule. The collector creates a simulated filesystem in the `<server-root>/logs` directory for each command type so that the Host System Monitor Provider can find the output of the most recently collected data.

Repeating commands use a subdirectory for each command type to keep results isolated from other command types and to help organize file cleanup. The filename of the output contains the sample timestamp, such as `iostats-[sampletimestamp]`. If the collector daemon fails for any reason, the Host System Monitor provider is not left reading stale system data because the expected timestamp files is missing. To handle clock-edge timing, the monitor sampler will also look for data in a filename of the previous second. Timestamp files are deleted once their data have been collected.

The collector daemon runs with no inter-process communication and can be stopped if no longer necessary.

Server clock skew

Server clock skew is the difference between the PingDataMetrics server system clock and the monitored server clock, in seconds.

Correlating metric samples from multiple servers requires that the timestamp associated with each sample from each monitored server is synchronized. The PingDataMetrics server tracks system time information and makes it visible in the `cn=Monitored Server <servername>,cn=monitor` entry.

The `system-clock-skew-seconds` attribute indicates the difference between the PingDataMetrics server system clock and the monitored server clock, in seconds. The larger this skew value, the less precision there is when comparing changes in data across servers.

While it is not necessary to keep the PingDataMetrics server clock synchronized with all of the monitored servers, it can be convenient when issuing metric queries with time ranges specified by offsets. Because the offset is computed using the PingDataMetrics server system clock, if this clock is very different from the monitored servers' system clocks, the start and end time of a metric query will not match the expected boundaries.

Tuning data collection

You can tune data collection by reducing the data collected, the frequency of data collection, the frequency of sample block creation, and PingDataMetrics server impact on performance.

Collecting all of the performance data at the most granular level from all of the servers might not be possible without a significant investment in hardware for the PingDataMetrics server. Instead, tune data collection to fit within the limits of the existing PingDataMetrics server hardware.

The remainder of this section describes several strategies for tuning data collection.

Reducing the data collected
If not all information collected by the PingDataMetrics server is required, the Stats Collector plugin's entry-cache property can be tuned using the `dsconfig` command-line tool.

The server collects information for eight different information groups. Limit data collection to the devices of actual interest.

To omit all metrics related to the entry cache, set the `entry-cache-info` group on the monitored server:

```
$ bin/dsconfig set-plugin-prop --plugin-name "Stats Collector" \
  --set entry-cache-info:none
```

Reducing the frequency of data collection
Use the `dsconfig` tool to reduce or modify the frequency of data collection in the monitored servers.

About this task

Monitored servers can produce metric samples every second, which is useful for short-duration changes. These samples are less useful hours later, after the per-second data is aggregated to per-minute data.

Steps

- To change the base sample production rate from the default of 1 second to 10 seconds, use the `dsconfig` tool.

  ```
  $ bin/dsconfig set-plugin-prop --plugin-name "Stats Collector" \
    --set "sample-interval:10 seconds"
  ```

  This change reduces the total data volume by about 90 percent.

Reducing the frequency of sample block creation
Reduce the number of sample blocks processed by the PingDataMetrics server.

About this task

By default, the monitored servers produce a new block of samples every 30 seconds. Increasing this to 60 seconds, while reducing the PingDataMetrics Server's polling rate to 60 seconds, reduces the sample processing overhead.

Steps

- To change the frequency at which the monitored servers create sample blocks, run the following `dsconfig` command:

  ```
  $ bin/dsconfig set-backend-prop --backend-name metrics \
    --set sample-flush-interval:60s
  ```

Reducing PingDataMetrics Server impact on performance

All Ping Identity servers expose performance data through the `cn=monitor` distinguished name (DN).

About this task

Performance issues occur when data is read, either directly by an LDAP client, or by enabling either the Stats Logger or Stats Collector plugins.

The Stats Logger plugin reads the configured monitors and writes the resulting values to a `.csv` file. The Stats Collector plugin also reads the configured monitors and writes the resulting values to a `.csv` file, but this file is made available for LDAP clients in `cn=metrics` DN. The Stats Collector `.csv` files are suitable for use by the PingDataMetrics server, and contain one metric value per line.

The Stats Logger and the Stats Collector plugins are both disabled by default. When enabled, each of these plugins adds an approximate 3% CPU utilization penalty, plus a negligible amount of disk I/O and Java virtual machine (JVM) heap usage.

Steps

- To enable the Stats Collector plugin, run the following `dsconfig` command:

```
$ bin/dsconfig set-plugin-prop --plugin-name "Stats Collector" \
  --set enabled:true
```

  The `monitored-servers` tool enables the Stats Collector plugin on the monitored server.

Data processing

When blocks of samples arrive in the PingDataMetrics server, they are queued on disk and loaded into the database.

Samples from a single server are processed in time-order, so that sample blocks with older data are always processed before a sample block containing newer data. The PingDataMetrics server does not do time-correlation between blocks coming from different servers. So, server A samples from two hours ago can be loaded immediately after server B samples from two minutes ago. This flexibility enables servers to be unavailable to the PingDataMetrics server, without affecting the overall system monitoring. Also, a query for data from server A and B can return data for server B but not server A, until the data queued for server A has been collected and imported. Samples collected from the PingDataMetrics server itself are processed ahead of all other servers.

Importing data

The PingDataMetrics server polls all of the monitored servers at a regular interval to collect data it to import into the database management system (DBMS).

When they are available, the PingDataMetrics server fetches new samples through LDAP. The PingDataMetrics server has one dedicated thread taking sample blocks and converting them to the normalized form stored in the DBMS. The import queue's size is normally near zero, but under certain conditions it can become large. When the PingDataMetrics server starts, it will queue for import all sample blocks still on disk. Blocks that are older than two hours are discarded.

> If a monitored server becomes unavailable for an extended period of time, it will continue to queue blocks of samples locally. When it becomes available again, the PingDataMetrics server collection poll of that server will capture hundreds or thousands of sample blocks. The PingDataMetrics server captures the sample blocks at a much faster speed than it can import them, causing the queue to grow for a period of time. If the PingDataMetrics server is stopped, this problem is compounded because all monitored servers will then have a backlog of sample blocks to be imported.

Aggregating data

To maintain a size-limited database management system (DBMS) while accumulating data over a period of years, the PingDataMetrics server aggregates data into four different levels.

The PingDataMetrics server aggregates data into four different levels. Each level contains data with less time granularity, but covering a larger period of time. Data is aggregated from a lower (greater time granularity) to a higher level as soon as enough data for aggregation is available. For example, the level 0 data has one second granularity, and the level 1 data has one minute granularity. After level 0 has collected one minute's worth of data, the data from that minute can be aggregated to level 1.

To keep the data tables for each aggregation level at a constrained size, each aggregation level has a maximum age for the samples. When the samples are older than this age, they are deleted from the level. While aggregation occurs soon after the samples arrive in the level, pruning occurs only after all samples in a block have passed their age limit.

The PingDataMetrics server attempts to collect data from all configured servers as efficiently as possible. However, monitored server availability, DBMS backlog, and PingDataMetrics server load can all cause the data pipeline to slow down. The data aggregation system is designed to correctly handle gaps in the data.

The resolution of the aggregation levels cannot be changed, but the maximum age of each level can be configured. The following table lists the aggregation levels.

Aggregation levels

Level	Resolution	Default maximum age	Maximum age
0	1 second	2 hours	48 hours
1	1 minute	7 days	34 days
2	1 hour	12 months	5 years
3	1 day	20 years	20 years

Monitoring for service level agreements

The PingDataMetrics Server provides the ability to aggregate and track performance data for one or more service level agreements (SLAs).

The server aggregates the data using an SLA object that tracks the current and historical performance of LDAP operations (throughput and response times) that are tied to specifically monitored applications. The SLA object consists of a tracked application name, one or more LDAP operations to be considered, a set of servers that contribute performance data to the SLA, and optionally, thresholds to generate alerts should the server exceed these limits.

Thresholds are optional configuration settings that enable the monitoring of performance data. Each threshold sets a limit that indicates a warning condition where the server's performance is nearing a limit and/or a critical condition. When the monitored server enters or ends a warning or critical state, the PingDataMetrics Server generates an alert. The generated alerts are the same as those created by the PingDirectory Server and PingDirectoryProxy Servers and can be routed through the Alert Handler to a monitoring console or administrator.

The SLA object can report the aggregate performance of all configured servers. The SLA object is configured with the following:

Designate servers that contribute to SLA tracking

The SLA object includes a Server Query component that is used to designate the servers that contribute to the SLA measurements.

REST API

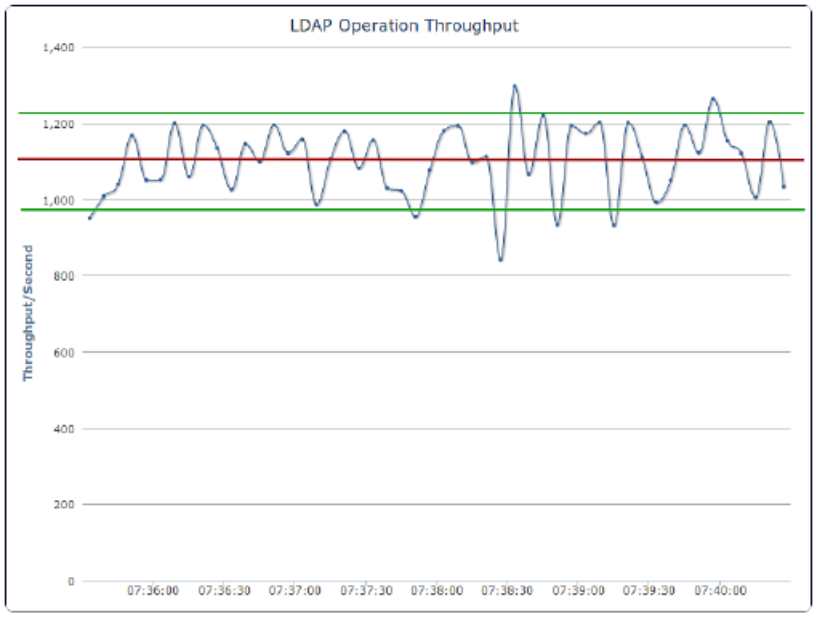
A REST API enables listing configured SLA objects and their current status. The PingDataMetrics Server REST API also enables listing alerts generated by SLA thresholds, and blending the alert information with the threshold information to provide a more contextual view of the tracked applications performance.

SLA thresholds

The PingDataMetrics Server uses a Monitoring Threshold mechanism that has two components.

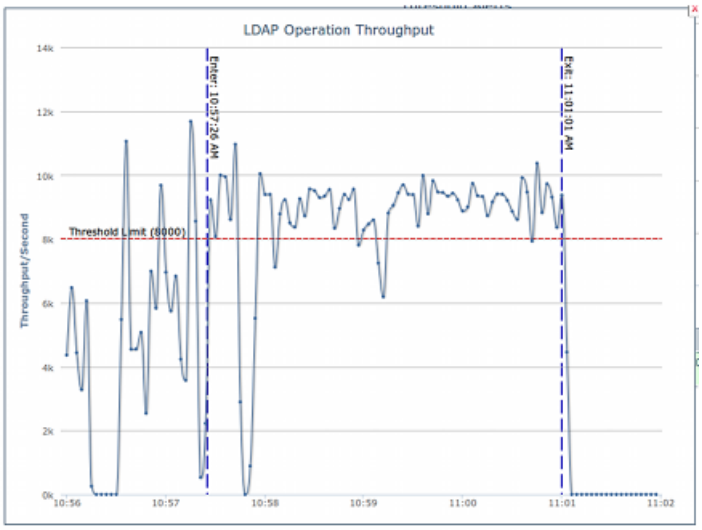
Spike monitoring threshold

Used to configure a set of operational performance limits on a specific measurement, where the limit is specified as a percent change from the most recent measurement average value. A Spike Monitoring Threshold has warning and critical limits, and will enter or leave an alerted state when the monitored value exceeds either of the limits. This threshold is useful when the valid range of the measurement is not known in advance. This type of limit is useful in detecting short-term changes in a measurement that fluctuates broadly over time. The limit is applied in both positive and negative directions, so that this type of threshold can detect an upward or downward spike in the value. The following chart shows a spike monitoring threshold, where the red line is the average throughput/second and the green lines are the limits, showing the average window for the throughput.



Static level monitoring threshold

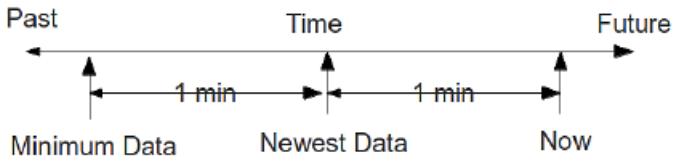
Used to configure performance limits on a specific measurement, where the specified limits are fixed values that do not change over time. A Static Level Monitoring Threshold has warning and critical limits, and will enter or leave an alerted state when the monitored value exceeds any of the limits. The Static Level Monitoring Threshold is configured with static numeric limits, and is useful when the expected valid range of the measurement is known in advance.



Threshold time line

The PingDataMetrics Server periodically evaluates each threshold, computing current value, current average, and alerted state.

The default evaluation period is 30 seconds. Using the figure below, a threshold is evaluated at time 'Now.' The most recent data that threshold uses is one minute old (Newest Data). Each threshold evaluation requires at least one minute of new data (Minimum Data). At time 'Now,' the threshold is working with data that is between one and two minutes old (between Minimum Data and Newest Data).



The one minute delay between 'Now' and 'Newest Data' is not configurable. This delay ensures that the PingDataMetrics Server has had enough time to poll the monitored servers and get the most recent data. The one minute delay between 'Newest Data' and 'Minimum Data' is configurable on a per-threshold basis for Spike-valued thresholds, but one minute is the minimum window. Generally, time between when a monitored performance anomaly occurs on a monitored system, and when an alert is created will be between two and three minutes

Because the PingDataMetrics Server can capture metrics data with a very fine time resolution (one second data is the default), the data is often very "noisy." By default, the data is time-averaged (using five consecutive one-second samples to produce a single five-second value), and time-averaging will ultimately reduce the noise. However, "noisy" data can make it harder to choose an appropriate threshold limit value. If the limit value is too close to the noise levels, the threshold will alert due to values that have a very short time duration.

Each threshold is configured with a `minimum-time-to-trigger` property, which determines the minimum time allowed to exceed the threshold before an alert is generated, and a `minimum-time-to-exist` property that determines the time required for the threshold to exit an alerted state.

Configuring an SLA object

Configure any number of SLA objects for monitored servers using the `dsconfig` command-line tool.

About this task

The service level agreement (SLA) object relies on existing performance metrics and only aggregates the data for specific SLAs. For more information, see the Ping Identity PingDataMetrics Server Configuration Reference (HTML) documentation in the *<server-root>*/docs directory.

Steps

1. Create a server query that specifies which servers will contribute to SLA monitoring.
   This command specifies the PingDirectoryProxy Servers located in Austin.

```
$ bin/dsconfig create-server-query \
  --query-name "Austin Proxy Servers" \
  --set server-instance-type:proxy \
  --set server-instance-location:Austin
```

2. Create a `static-level` monitoring threshold.
   In this example, the alert condition is set to `entry`, which means that the server will generate an alert if the server enters a warning state (`alert-on-warn:true` and `warn-if-above:12`) or critical state (`critical-if-above:15`). When the server leaves its alerted state, an alert is generated (`alertcondition: exit`). The minimum amount of time that the threshold can be exceeded before an alert is generated is set to 15 seconds (`min-time-for-trigger:15s`).

```
$ bin/dsconfig create-monitoring-threshold \
  --threshold-name "15ms response time" \
  --type static-level \
  --set alert-condition:entry \
  --set alert-condition:exit \
  --set alert-on-warn:true \
  --set min-time-for-trigger:15s \
  --set min-time-for-exit:15s \
  --set warn-if-above:12 \
  --set critical-if-above:15
```

3. Create another `static-level` monitoring threshold.
   For this example, the alert condition is set to `entry`. The server generates an alert if it enters a warning state (`alert-on-warn:true` and `warn-if-above:4000`) or critical state (`critical-if-above:5000`). When the server leaves its alerted state, an alert is generated (`alertcondition:exit`). The minimum amount of time that the threshold can be exceeded before an alert is generated is set to 15 seconds.

```
$ bin/dsconfig create-monitoring-threshold \
  --threshold-name "5k ops/sec" \
  --type static-level \
  --set alert-condition:entry \
  --set alert-condition:exit \
  --set alert-on-warn:true \
  --set min-time-for-trigger:15s \
  --set min-time-for-exit:15s \
  --set warn-if-above:4000 \
  --set critical-if-above:5000
```

4. Create an SLA object that targets a single sign-on (SSO) application and monitors the response and throughput times for LDAP bind operations.
   The response time threshold is set to `15ms`. The throughput threshold is set to `5k` operations per second. The targeted servers are the set of PingDirectoryProxy Servers, located in Austin.

```
$ bin/dsconfig create-ldap-sla \
  --sla-name "SSO Application" \
  --set enabled:true \
  --set "application-name:SSO Application" \
  --set "response-time-threshold-ms:15ms response time" \
  --set "throughput-threshold-ops-per-second:5k ops/sec"
  --set ldap-op:bind \
  --set "sla-server-query:Austin Proxy Servers"
```

Configuring charts and dashboards

The PingDataMetrics Server provides a set of dashboards with series of charts for each configured PingData server.

Charts can be built and customized with the PingDataMetrics Server Chart Builder tool. Dashboards and charts can be modified with Velocity templates.

Available dashboards

The PingDataMetrics Server includes several dashboards that can be used to display information for all servers in a data center, specific applications, or service level agreement (SLA) specifics.

The following dashboards are available:

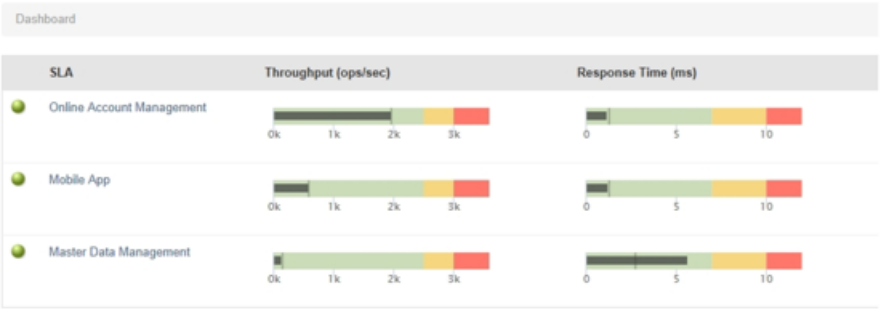
ldap-dashboard

Displays charts for PingDirectory Server, PingDirectoryProxy Server, and PingDataSync Servers configured with the monitored-servers command. Charts are also displayed for the PingDataMetrics Server server. This dashboard is viewed from a browser at `http://<metrics-host>:<port>/view/ldap-dashboard`, and is easily customized. For more information, see *Customize the LDAP dashboard*. The charts can display information by:

- Individual server, server location, or server type.
- Varying level of detail adjusted by server type.
- Time scale, providing either a recent or more historical data view.

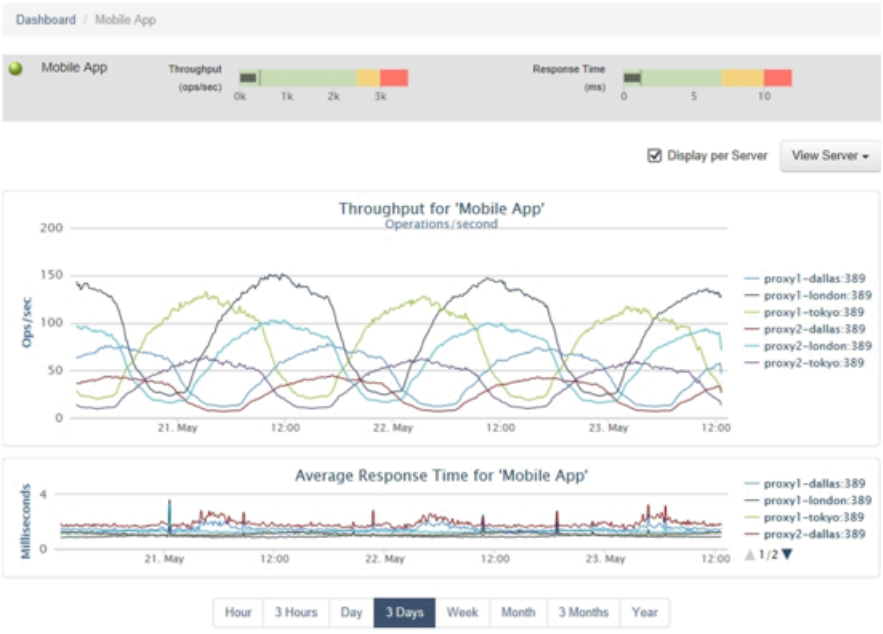
sla-viewer

Displays throughput and response time graphs, and status for configured SLAs. This dashboard is viewed from a browser at `http://<metrics-host>:<port>/view/sla-viewer`. See *Monitoring for Service Level Agreements* for information about configuring SLAs.



sla-viewer-details

Displays SLA Viewer data and additional charts for response time and time ranges from the sla-viewer dashboard. Data can be viewed per server and includes server details.



demo-dashboard

Demonstrates how to display a set of charts for multiple servers and how to vary that set of charts per server type. This dashboard is viewed from a browser at `http://<metrics-host>:<port>/view/demo-dashboard`. You can use the dashboard as a starting point for custom dashboards.

A dashboard `.readme` file provides general instructions for customizing any dashboard, and is located in `<server-root>/config/dashboard/dashboard.README`.

You can create and reference custom style sheets in the dashboard template or configure styles for all charts. See *Chart Presentation Details* for information. The PingDataMetrics Server default style sheets should not be modified.

Customizing the LDAP dashboard

Use configuration files in the Velocity templates to customize the LDAP dashboard.

Dashboards are defined by Velocity templates. After servers are configured, the LDAP dashboard displays all metrics from monitored servers. See *Velocity templates* for more information about templates and template components.

Use the following options and resources to configure and customize the LDAP dashboard:

- Access the configuration file for the LDAP dashboard at `<server-root>/config/velocity/templates/_ldap-dashboard-config.vm`.

  > **Note:**
  >
  > Use the `config` file as a guide for customization, but do not change it.
  >
  > Files within this directory that begin with an underscore (_) are templates that are referenced by each of the dashboards. The `_ldap-dashboard-config.vm` template is the only file that contains all of the dashboard configuration inside the file. Configuration of other templates requires configuration of a corresponding dashboard file as well.

- The `_ldap-dashboard-config.vm` file references the customizable template file in `<server-root>/velocity/templates/_ldap-dashboard-config-overrides.vm.`. Use this file to perform customizations.
- Both the `_ldap-dashboard-config.vm` and `_ldap-dashboard-config-overrides.vm` files contain configuration instructions.
- You can customize the following in the `_ldap-dashboard-config-overrides.vm` file:
  - The charts that display for each server type and their styles. See *Available Server Charts* for more information.
  - The charts that display for a data center and their styles.
  - The charts that display for an application type and their styles.
  - The default time resolution (two weeks, is the default for data displayed).
  - The size of the charts.

Debug dashboard customization

Use a debug option in any Velocity template for exploring available information in the Velocity Context.

This information includes the servers that are monitored and the metrics that are available. This option is included in the `ldap-dashboard` and `demo-dashboard` files:

```
## Uncomment this to have a window popup with detail of what's in the
 Velocity
Context.
##parse("_debug.vm")
##debug()
```

See *Velocity templates* on page 1605 for more information.

Preserve customized files

Any files that are customized should be copied from the `config/velocity` subdirectories to the same subdirectory of the velocity directory under the server root (`<serverroot>/velocity`).

> **Note:**
>
> The files in `config/velocity` should not be modified. They are updated when the product is updated.

By default, any file of the same name under <server-root>/velocity will be loaded in place of `<server-root>/config/velocity`. This enables the preservation of customized files after a product upgrade.

> ⓘ **Note:**
>
> After a product upgrade, review the files in `config/velocity` to determine if any changes should be incorporated into customized templates.

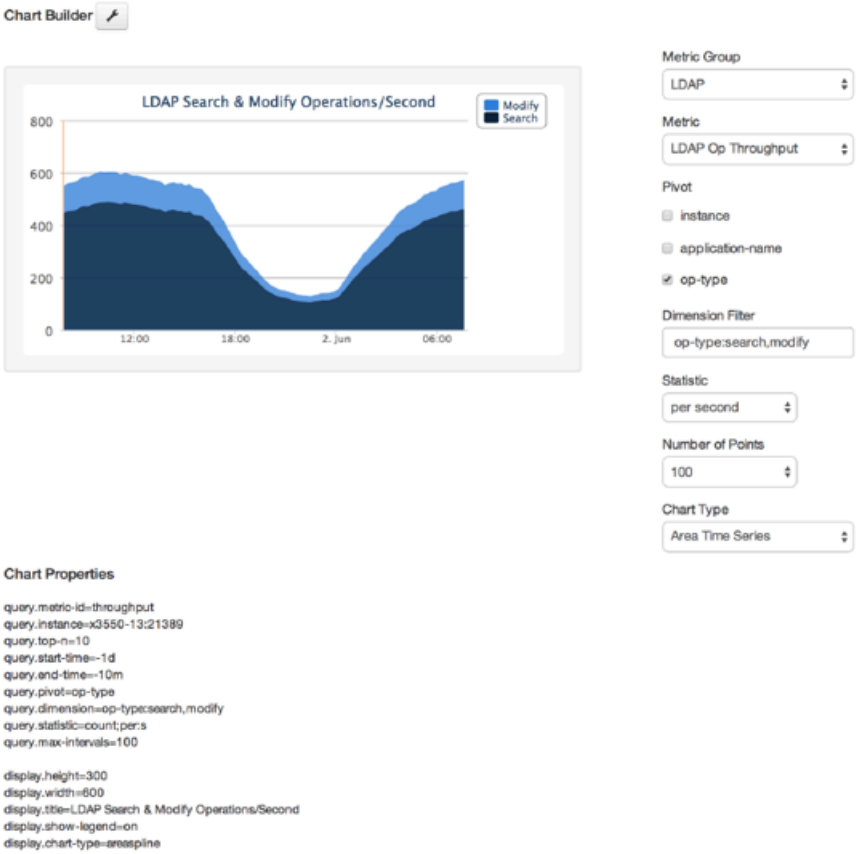
The Chart Builder tool

The Chart Builder tool is used to create performance charts for all configured servers.

The Chart Builder tool (`chart-builder.vm`) is shipped with the PingDataMetrics Server and is enabled after installation at the following URL: `https://<metricshost>:<port>/view/chart-builder`.

As the settings in the Chart Builder are changed, the builder gathers the data from the PingDataMetrics Server using the PingDataMetrics Server REST API. Once configured, the dashboard page asynchronously fetches metric data for all charts, with each chart rendering when its data is returned. While most metric queries respond quickly (50-100ms), some queries can take longer. If the lag seems too long, consider making changes to the query to reduce the amount of data gathered.

Selecting specific instances and using dimension filters can decrease query time. The Chart Builder tool and the underlying libraries constrain a chart to a single metric. The size of each chart is determined by the library default size (300x300) and can be overridden in the chart properties file. There are times when the legends and labeling of a chart dictate the minimum size for a chart.



The metrics parameters used to build the chart can be saved to a properties file and added to a dashboard. If not using the Chart Builder tool, the `_chart-definition.template` file in *<server-root>*`/config/dashboard/charts` provides instructions about manually creating charts and adding them to a dashboard.

Chart presentation details
You can configure chart presentation details per chart or for all charts in the `_chart-definition.template` file.

A properties file can be created for common styles and referenced in this file. Instructions for adding custom styles are included in the file. The following is a sample of the chart parameters that are available:

- Colors used in the data series
- Enable and disable a legend
- Location (top/bottom/left/right) of the legend
- Background color
- Thickness of the time-series lines (absolute or as a function of the # of plotted series)
- Macro expansion in the specified title
- Sub-title (with macro expansion that includes metric-name and current date/time)

Chart Builder parameters
Refer to a table of the available Chart Builder parameters for the PingDataMetrics server.

Use the Chart Builder tool to build or adjust system and performance charts. When the configuration is set, copy the parameters into a properties file, and add the chart to a dashboard.

Chart Builder parameters

Parameters	Description
Metric Group	Selects a specific group of metrics to be considered for charting.
Metric	Displays the specific metric. Open the drop-down list and hover over a metric to view a description of the particular metric.
Pivot	Splits the chart result into multiple series based on the pivot dimension chosen.
Dimension Filter	Filters the data based on the dimension(s) entered, such as the type of operations that can be viewed for an LDAP operations metric.
Statistic	Displays the type of "measurement" that may exist for each metric. For example, each response-time sample contains: ▪ number of operations (count) ▪ average time-per-op (average) ▪ histogram-of-operation-time (histogram) On a per-sample basis, the PingDataMetrics server stores the following: count, average, minimum, maximum, and histogram. Any metric can have one or more of five statistics, but not all statistics are equally valuable. Note the following points: ▪ The minimum and maximum statistics may be of limited value, because as they are time-averaged, they go to extremes (min of minimums and max of maximums). ▪ The count and histogram statistics have high fidelity over time because they time-aggregate perfectly. ▪ The average statistic loses fidelity over time, because as the time-window for averaging gets larger, the highs and lows are lost.

Parameters	Description
Number of Points	If the number of points is set to 1, all chart types, except time series, can be used. If the number of points is greater than 1, then only time series charts can be used.
Chart Type	Displays the chart based on the type: ▪ Area time series ▪ Bar chart ▪ Column chart ▪ Pie chart ▪ Stacked bar chart ▪ Stacked column chart ▪ Time series
Chart Properties	Displays the generated chart properties. Copy the query properties into a properties file.
Data URL	The API for getting the data. This can be used to call the chart into a third-party client application.

Chart properties file
Each dashboard uses a Velocity template (*<name>*`.vm`) and a set of chart properties files to render the charts.

As charts are configured with the Chart Builder tool, the tool generates the corresponding properties for each customized item. The metrics configuration can be copied into a properties file and added to a dashboard. If no values are specified for a given property, the property will use a default value from the *<server-root>*`/config/dashboard/charts/_chart-definition.template` file. All properties and their descriptions are listed in this file.

The properties in the chart definition file are broken into two groups:

- Properties that start with `display` affect the display of the data.
- Properties that start with `query` affect the metric query.

When building a new chart, just copy the query parameters into a properties file. In general, display options should be referenced from common styles defined in the `_chart-definition.template` file, or the styles defined for a dashboard.

Available charts for PingData servers

The following are the default charts that display on the LDAP Dashboard for each configured server. These and additional charts for server and system metrics reside in *<server-root>*`/config/dashboard/charts`. They can be modified or used to create new charts.

Charts for all servers

The following charts are displayed on the LDAP Dashboard for PingDirectory Server and PingDirectoryProxy Servers:

- LDAP Read Operations Per Second
- LDAP Write Operations Per Second
- LDAP Response Time
- LDAP Response Time Outliers
- LDAP Worker Thread Percent Busy
- LDAP Average Operations in Progress

- LDAP Average Queue Size
- LDAP Open Connections
- LDAP New Connections
- System CPU
- System Memory Percent Free
- System Network Read MB
- System Network Write MB
- System Disk Busy
- System Disk Service Wait
- System Disk Read MB
- System Disk Write MB

PingDirectory Server charts

The following charts are displayed on the dashboard for the PingDirectory Server:

- Replication Backlog
- Replication Oldest Change
- Replication Unresolved Naming Conflicts
- Backend Entry Count
- Backend Cache Percent Full
- Backend Size on Disk
- Backend Cleaner Backlog

PingDirectoryProxy Server charts

The following charts are displayed on the dashboard for the PingDirectoryProxy Server:

- External Server Total Operations
- External Server Failed Operations
- External Server Health

PingDataSync Server charts

The following charts are displayed on the dashboard for the PingDataSync Server:

- Sync Pipe Unretrieved Changes
- Sync Pipe Percent Busy
- Sync Pipe Completed Operations Success
- Sync Pipe Completed Operations Failed
- Sync Pipe Completed Operations By Type

PingDataMetrics Server charts

The following charts are displayed on the dashboard for the PingDataMetrics Server:

- Metrics Queries Per Minute
- Metrics Query Time
- Metrics Query Time Max
- Metrics Query Time Histogram
- Metrics Cache Entry Count
- Metrics Cache Hit Count
- Metrics Cache Miss Count
- Metrics Cache Expired Count
- Metrics Cache Evicted Count
- Metrics Import Delay
- Metrics Load Time

- Metrics DBMS Cluster Time

PingDataGovernance Server charts

The following charts are displayed on the dashboard for the PingDataGovernance Server:

- Policy Evaluation
- Policy Evaluation Time

Velocity templates

Velocity template files contain presentation content and variables that are replaced when the content is requested.

The PingDataMetrics Server exposes Velocity pages through an HTTP servlet extension. If the HTTP Connection Handler is enabled, the Velocity extension is enabled.

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

Variables are expressed using a `$` followed by an identifier that refers to an object put into a context (VelocityContext) by the server.

`expose-*` properties

Velocity extensions can be configured to expose a number of objects in the context using the `expose-*` properties:

`expose-request-attributes`

Indicates whether HTTP request attributes are accessible to templates using the `$ubid_request` variable. In general, request attributes are added by server components processing the HTTP request. Also the HTTP request parameters map is available as `$ubid_request.parameters`. Request parameters are supplied by the requester, usually in the request URL query string or in the body of the request itself.

`expose-session-attributes`

Indicates whether HTTP session attributes are accessible to templates using the `$ubid_session` variable. Like request attributes, session attributes are also added by server components processing the HTTP request. The lifetime of these attributes persists until the user's session has ended.

`expose-server-context`

Indicates whether a Server SDK server context is accessible to templates using the `$ubid_server` variable. The server context provides access to properties and additional information about the server. See the Server SDK documentation for details.

Velocity HTTP servlet extension properties

The following are other properties of the Velocity HTTP servlet extension:

`description`

A description of the extension.

`cross-origin-policy`

Defines a cross origin policy for this extension.

`base-context-path`

URL base context for the Velocity Servlet.

`static-content-directory`

In addition to templates, the Velocity Servlet serves miscellaneous static content related to the templates. By default this is `config/velocity/statics`.

`require-authentication`

Requires credentials to access Velocity content.

`identity-mapper`

Maps user credentials to backend entries. If the `require-authentication` property is set, use this property to map bind credentials from a users backend. This is set to Exact Match by default. PingDataMetrics Server Velocity template authentication should share the api-users LDIF backend used by the REST API. Details are available in the PingDataMetrics Server REST API servlet configuration, and in the *Connection and Security* section.

`static-custom-directory`

If static content is customized, it resides in `velocity/statics` by default.

`template-directory`

The template directory from which templates are read. By default this is `config/velocity/templates`. This directory also serves as a default for Template Loaders that do not have a template directory specified.

`static-context-path`

URL path beneath the base context where static content can be accessed.

`allow-context-override`

Indicates whether context providers can override existing context objects with new values.

`mime-types-file`

Specifies a file that is used to map file extensions of static content to a Content Type to be returned with requests.

`default-mime-type`

The default Content Type for HTTP responses. Additional content types are supported by defining one or more additional Velocity Template Loaders.

The VelocityContext object can be further customized by configuring additional Velocity context providers. The dot notation used for context references can be extended to access properties and methods of objects in context using Java Bean semantics. For example, if the HTTP request URL includes a name query string parameter like:

```
http://example.com:8080/view/hello?name=Joe
```

An HTML template like the following could e used to generate a page containing a friendly greeting to the requestor:

```
<html>
  <body>
    Hello $ubid_request.parameters.name
  </body>
</html>
```

A pop-up window displays a table on the page that lists all variables that are in the Velocity Context. References like `$ubid_request` can appear in the template file and be replaced when the template is rendered. This information can be used to check which variables are permitted to be in the template along with the variable values.

> ⓘ **Note:**
>
> For security, all template substitutions are HTML escaped by default. To substitute unescaped content, you must use a variable name ending with `WithHtml`. For example, `$addressWithHtml`, would substitute the contents of the `$addressWithHtml` variable into the page generated from the HTML template without escaping it.

A debug option can be used in any Velocity template for verifying available information in the Velocity Context:

```
parse("_debug.vm")
debug()
```

If a variable is added to a template for something that does not exist, the rendered page will contain a literal string of the unfulfilled variable, such as `$undefined_variable`.

By default, the Velocity Servlet Extension expects to access content in subdirectories of the server's `config/velocity` directory:

templates

This directory contains Velocity template files that are used to generate pages in response to client requests.

statics

This directory contains static content, such as CSS, HTML, and Javascript files as well as images and third-party libraries.

Supporting multiple content types

Velocity can be configured to support HTTP requests with different content types.

By default, the Velocity Servlet Extension is configured to respond to HTTP requests with a content type `text/html`. Change this request type by setting the default MIME type using `dsconfig`. For example, the following can be used to set the default type to XML:

```
$ bin/dsconfig set-http-servlet-extension-prop \
  --extension-name Velocity \
  --set default-mime-type:application/xml
```

HTML requests can be supported as well as clients that seek content in other formats. Create one or more Velocity template loaders to load templates for other content types like XML or JSON.

The ability to serve multiple formats of a document to clients at the same URL is typically called "content negotiation". HTTP clients indicate the type of content desired using the `Accept` header. A client may use a header like the following to indicate that they prefer content in XML but will fallback to HTML if necessary:

```
Accept:application/xml,text/html;q=0.9
```

The following can be used to create a Velocity template loader for XML content:

```
$ bin/dsconfig create-velocity-template-loader \
  --extension-name Velocity \
  --loader-name XML \
  --set evaluation-order-index:502 \
  --set mime-type-matcher:application/xml \
  --set mime-type:application/xml \
  --set template-suffix:.vm.xml
```

Upon receiving a request, the Velocity Servlet first creates an ordered list of requested media types from most desired to least based on the value of the `Accept` header. Starting from the most desired type, it will then iterate over the defined template loaders according to the `evaluation-order-index` property from lowest value to highest.

A template loader can indicate that it can handle content for requested media type by comparing the requested type to its `mime-type-matcher` property. A loader can be configured to load templates from a specific directory or load template files having a particular suffix. For example, XML templates are expected to be named using a `.vm.xml` suffix. If a loader indicates it handles the requested content type and a template exists for the requested view, the template is loaded and used to generate a response to the client. If no loaders are found for the requested media type, the next most preferred media type (if any) is tried. If no loaders indicated that they could satisfy the requested view, the client is sent an `HTTP 404 (not found)` error. If no loaders could provide acceptable media but the requested view exists in some other format, the client is sent an `HTTP 406 (not acceptable)` error.

In this example, a template file called hello.vm.xml can be used to generate a response in XML:

For this example, a template file called `hello.vm.xml` can be used to generate a response in XML:

```
<hello name="$ubid_request.parameters.name"/>
```

In this case, the response will contain an HTTP Content-Type header with the value of the `mime-type` property of the Velocity template loader.

Velocity context providers

Velocity Context Providers allow more flexibility when populating the Velocity Context for template use.

The previous examples use a value supplied as an HTTP request query string parameter to form a response. The templates contain a variable `$ubid_request.parameters.name` that was replaced at runtime with a value from the Velocity Context.

The Velocity Extension can be configured to make some information available in the Velocity Context such as the HTTP request, session, and Server SDK Server Context.

Here are some of the properties of a Velocity Context Provider:

enabled

Indicates whether the provider will contribute content for any requests.

object-scope

Indicates to the provider how often objects contributed to the Velocity Context should be re-initialized. Possible values include `request`, `session`, or `application`.

`included-view`, `excluded-view`

These properties can be used to restrict the views for which a provider contributes content. A view name is the request URL's path to the resource without the Velocity Servlet's context or a leading forward slash. If one or more views are included, the provider will service requests for just the specified views. If one or more views are excluded, the provider will service requests for all but the excluded views.

Velocity Tools context provider

The Velocity context provides access to a set of Velocity Tools.

Apache's Velocity Tools project is focused on providing utility classes useful in template development. The Velocity Context can be configured by specifying Velocity Tool classes to be automatically added to the Velocity Context for template development. For more information about the Velocity Tools project, see the Velocity website.

You can use the following command to list the set of Velocity Tools that are included in the Velocity Context for general use by templates:

```
$ bin/dsconfig get-velocity-context-provider-prop \
  --extension-name Velocity \
  --provider-name "Velocity Tools" \
  --property request-tool \
  --property session-tool \
  --property application-tool \
```

Troubleshooting

There are several ways to troubleshoot issues with data gathering or with the PingDataMetrics Server itself.

Using the collect-support-data tool

Run the `collect-support-data` tool to aggregate data and send it to a support provider.

About this task

PingData servers provide information about their current state and any problems encountered. If a problem occurs, run the `collect-support-data` tool in the `/bin` directory. The tool aggregates all relevant support files into a `.zip` file that can be sent to a support provider for analysis. The tool also runs data collector utilities, such as `jps`, `jstack`, and `jstat` plus other diagnostic tools for the operating system.

The tool can only archive portions of certain log files to conserve space, so that the resulting support archive does not exceed the typical size limits associated with e-mail attachments.

The data collected by the `collect-support-data` tool can vary between systems. The data collected includes the configuration directory, summaries and snippets from the logs directory, an LDIF of the monitor and RootDSE entries, and a list of all files in the server root.

Perform the following steps to run this tool:

Steps

1. From the server root directory, run the `collect-support-data` tool.
   Include the host, port number, bind DN, and bind password.

   ```
   $ bin/collect-support-data \
     --hostname 100.0.0.1 --port 389 \
     --bindDN "cn=Directory Manager"
     --bindPassword secret \
     --serverRoot /opt/PingData<server> \
     --pid 1234
   ```

2. Email the `.zip` file to a support provider.

Slowing queries based on sample cache size

Increase the sample cache size and the idle timeout using the `dsconfig` tool.

About this task

The `evicted-count` attribute of the sample cache sets the number of entries that have been evicted from the cache due to a lack of space. The cache might not be large enough for the query load placed on the server.

Steps

1. To increase the size of the sample cache to the maximum size of `200000`, run the following command:

```
$ dsconfig set-monitoring-configuration-prop \
  --set sample-cache-max-cached-series:200000
```

> ⓘ **Note:**
>
> Some queries are so infrequent that the cached data expires due to age. The default age is 10 minutes, but this can be increased up to one hour.

2. Optional: If the `expired-count` monitor attribute is increasing between queries, consider increasing the idle timeout with the command:

```
$ dsconfig set-monitoring-configuration-prop \
  --set sample-cache-idle-series-timeout:20m
```

Troubleshooting insufficient memory errors

Increase the heap size, or reduce the number of request handler threads using the `dsconfig` tool to help resolve insufficient memory errors.

About this task

If the server shuts down due to insufficient memory errors, it is possible that the allocated heap size is not enough for the amount of data being returned.

Steps

- To increase the heap size or reduce the number of request handler threads, run the command:

```
$ bin/dsconfig set-connection-handler-prop \
  --handler-name "HTTP Connection Handler" \
  --set num-request-handlers:<num-of-threads>
```

Unexpected query results

The samples of various servers, especially those with clock skew, can return unexpected query results.

The query API aggregates data samples across servers and dimension values. The samples for different servers, or even different dimension values, are imported into the PingDataMetrics Server at different times. All metric data is imported in time-order for each server. The ordering cannot be set across servers, and samples for a specific time can arrive in stages. Therefore, a metric query that aggregates across servers or dimensions might get partial data when the query time range ends. This problem can be compounded when the monitored servers clocks are not synchronized (samples have the monitored server timestamp). The query looks at a single time range. The more clock skew between the monitored servers, the higher the probability of the results not being accurate for the range.

With the query API, the data can be pivoted (split) by server and dimension. The API enables formatting the results as an HTML table. The following sequence of API URLs return the last three minutes of data in 10-second increments:

```
http://<metrics-server-host:port>/api/v1/metrics/throughput/datatable?
maxIntervals=30&startTime=-3m&tqx=out:html&tz=US/Central

http://<metrics-server-host:port>/api/v1/metrics/throughput/datatable?
maxIntervals=30&startTime=-3m&tqx=out:html&tz=US/Central&pivot=instance
```

```
http://<metrics-server-host:port>/api/v1/metrics/throughput/datatable?
maxIntervals=30&startTime=-
3m&tqx=out:html&tz=US/Central&pivot=instance&pivot=op-type
```

- The first URL aggregates all servers and LDAP operations into a single number split across time.
- The second URL splits out the data by server and time.
- The third URL splits out the data by server, LDAP operation, and time.

> ⓘ **Note:**
>
> As dimension pivots (splits) are added, the results display more aggregations of partial data.

Conditions for automatic server shutdown

Certain conditions will cause the PingData servers to shut down.

All PingData servers will shut down if any of the following occur:

- Server runs out of memory condition
- Low disk space error state
- Server runs out of file descriptors

Configuring the PingDirectory Server to shut down instead of lock down

- The PingDirectory Server will enter lockdown mode on unrecoverable database environment errors, but can be configured to shutdown instead with this setting:

```
$ dsconfig set-global-configuration-prop \
  --set unrecoverable-database-error-mode:initiate-server-shutdown
```

Troubleshooting installation and maintenance issues

The following topics include common installation and maintenance issues and possible solutions.

The setup program will not run
If the `setup` tool does not run properly, reference a list of common reasons and their solutions.

If the `setup` tool does not run properly, some of the most common reasons include:

A Java environment is not available

The server requires that Java be installed on the system prior to running the `setup` tool.

If there are multiple instances of Java on the server, run the setup tool with an explicitly-defined value for the `JAVA_HOME` environment variable that specifies the path to the Java installation. For example:

```
$ env JAVA_HOME=/ds/java ./setup
```

Another issue may be that the value specified in the provided JAVA_HOME environment variable can be overridden by another environment variable. If that occurs, use the following command to override any other environment variables:

```
$ env UNBOUNDID_JAVA_HOME="/ds/java" UNBOUNDID_JAVA_BIN="" ./setup
```

Unexpected arguments provided to the JVM

If the `setup` tool attempts to launch the Java command with an invalid set of arguments, it can prevent the Java virtual machine (JVM) from starting. By default, no special options are provided to the JVM

when running setup, but this might not be the case if either the `JAVA_ARGS` or `UNBOUNDID_JAVA_ARGS` environment variable is set. If the `setup` tool displays an error message that indicates that the Java environment could not be started with the provided set of arguments, run the following command:

```
$ unset JAVA_ARGS UNBOUNDID_JAVA_ARGS
```

The server is already configured or started

The `setup` tool is only intended to provide the initial configuration for the server. It will not run if it detects that it has already been run.

A previous installation should be removed before installing a new one. However, if there is nothing of value in the existing installation, the following steps can be used to run the `setup` program:

1. Remove the `config/config.ldif` file and replace it with the `config/update/config.ldif.{revision}` file containing the initial configuration.
2. If there are any files or subdirectories in the `db` directory, then remove them.
3. If a `config/java.properties` file exists, then remove it.
4. If a `lib/setup-java-home` script (or `lib\set-java-home.bat` file on Microsoft Windows) exists, then remove it.

The server will not start

There are various reasons why the server will not start.

If the server does not start, then there are a number of potential causes.

The server or other administrative tool is already running

Only a single instance of the server can run at any time from the same installation root. Other administrative operations can prevent the server from being started. In such cases, the attempt to start the server should fail with a message like:

```
The <server> could not acquire an exclusive lock on file
/ds/PingData<server>/locks/server.lock:
The exclusive lock requested for file
/ds/PingData<server>/locks/ server.lock
was not granted, which indicates that another
process already holds a shared or exclusive lock on
that file. This generally means that another instance
of this server is already running.
```

If the server is not running (and is not in the process of starting up or shutting down), and there are no other tools running that could prevent the server from being started, it is possible that a previously-held lock was not properly released. Try removing all of the files in the locks directory before attempting to start the server.

There is not enough memory available

When the server is started, the Java virtual machine (JVM) attempts to allocate all memory that it has been configured to use. If there is not enough free memory available on the system, the server generates an error message indicating that it could not be started.

There are a number of potential causes for this:

- If the amount of memory in the underlying system has changed, the server might need to be re-configured to use a smaller amount of memory.
- Another process on the system is consuming memory and there is not enough memory to start the server. Either terminate the other process, or reconfigure the server to use a smaller amount of memory.

- The server just shut down and an attempt was made to immediately restart it. If the server is configured to use a significant amount of memory, it can take a few seconds for all of the memory to be released back to the operating system. Run the `vmstat` command and wait until the amount of free memory stops growing before restarting the server.
- If the system is configured with one or more memory-backed filesystems (such as `/tmp`), determine if any large files are consuming a significant amount of memory. If so, remove them or relocate them to a disk-based filesystem.

An invalid Java environment or JVM option was used

If an attempt to start the server fails with '`no valid Java environment could be found`,' or 'the Java environment could not be started,' and memory is not the cause, other causes may include the following:

- The Java installation that was previously used to run the server no longer exists. Update the `config/java.properties` file to reference the new Java installation and run the `bin/dsjavaproperties` command to apply that change.
- The Java installation has been updated, and one or more of the options that had worked with the previous Java version no longer work. Re-configure the server to use the previous Java version, and investigate which options should be used with the new installation.
- If an `UNBOUNDID_JAVA_HOME` or `UNBOUNDID_JAVA_BIN` environment variable is set, its value can override the path to the Java installation used to run the server (defined in the `config/java.properties` file). Similarly, if an UNBOUNDID_JAVA_ARGS environment variable is set, then its value might override the arguments provided to the JVM. If this is the case, explicitly unset the `UNBOUNDID_JAVA_HOME`, `UNBOUNDID_JAVA_BIN`, and `UNBOUNDID_JAVA_ARGS` environment variables before starting the server.

Any time the `config/java.properties` file is updated, the `bin/dsjavaproperties` tool must be run to apply the new configuration. If a problem with the previous Java configuration prevents the `bin/dsjavaproperties` tool from running properly, remove the `lib/set-java-home` script (or `lib\set-java-home.bat` file on Microsoft Windows) and invoke the `bin/dsjavaproperties` tool with an explicitly-defined path to the Java environment, such as:

```
$ env UNBOUNDID_JAVA_HOME=/ds/java bin/dsjavaproperties
```

An invalid command-line option was used

There are a small number of arguments that can be provided when running the `bin/start-server` command. If arguments were provided and are not valid, the server displays an error message. Correct or remove the invalid argument and try to start the server again.

The server has an invalid configuration

If a change is made to the server configuration using dsconfig or the Administrative Console, the server will validate the change before applying it. However, it is possible that a configuration change can appear to be valid, but does not work as expected when the server is restarted.

In most cases, the server displays (and writes to the error log) a message that explains the problem. If the message does not provide enough information to identify the problem, the `logs/config-audit.log` file provides recent configuration changes, or the `config/archivedconfigs` directory contains configuration changes not made through a supported configuration interface. The server can be started with the last valid configuration using the `--useLastKnownGoodConfig` option:

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

To determine the set of configuration changes made to the server since the installation, use the `config-diff` tool with the arguments `--sourceLocal --targetLocal --sourceBaseline`. The `dsconfig --offline` command can be used to make configuration changes.

Proper permissions are missing

The server should only be started by the user or role used to initially install the server. However, if the server was initially installed as a non-root user and then started by the root account, the server can no longer be started as a non-root user. Any new files that are created are owned by root.

If the user account used to run the server needs to change, change ownership of all files in the installation to that new user. For example, if the server should be run as the "ds" user in the "other" group, run the following command as root:

```
$ chown -R ds:other /ds/PingData<server>
```

The server has shut down
There are several reasons why the server can shut down.

Check the current server state by using the `bin/server-state` command. If the server was previously running but is no longer active, potential reasons can include the following.

- Shut down by an administrator – Unless the server was forcefully terminated, then messages are written to the error and server logs stating the reason.
- Shut down when the underlying system crashed or was rebooted – Run the `uptime` command on the underlying system to determine what was recently started or stopped.
- Process terminated by the underlying operating system – If this happens, a message is written to the system error log.
- Shut down in response to a serious problem – This can occur if the server has detected that the amount of usable disk space is critically low, or if errors have been encountered during processing that left the server without worker threads. Messages are written to the error and server logs (if disk space is available).
- Java virtual machine (JVM) has crashed – If this happens, then the JVM should provide a fatal error log (a `hs_err_pid<processID>.log` file), and potentially a core file.

The server will not accept client connections
If the server is not accepting connections, there are several reasons this can happen.

Check the current server state by running the `bin/server-state` command. If the server does not appear to be accepting connections from clients, reasons can include the following:

- The server is not running.
- The underlying system on which the server is installed is not running.
- The server is running, but is not reachable as a result of a network or firewall configuration problem. If that is the case, connection attempts should time out rather than be rejected.
- If the server is configured to allow secure communication through SSL or StartTLS, a problem with the key manager or trust manager configuration can cause connection rejections. Messages are written to the server access log for each failed connection attempt.
- The server may have reached its maximum number of allowed connections. Messages should be written to the server access log for each rejected connection attempt.
- If the server is configured to restrict access based on the address of the client, messages should be written to the server access log for each rejected connection attempt.
- If a connection handler encounters a significant error, it can stop listening for new requests. A message should be written to the server error log with information about the problem. Restarting the server can also solve the issue. Another option is to create an LDIF file that disables and then re-enables the connection handler, create the `config/auto-process-ldif` directory if it does not already exist, and then copy the LDIF file into it.

The server is unresponsive

Check to see if the server is responsive with the `bin/server-state` command. If the server is not responding, there are various reasons and solutions to diagnose and fix this issue.

Check the current server state by using the `bin/server-state` command. If the server process is running and appears to be accepting connections but does not respond to requests received on those connections, potential reasons for this can include the following.

- If all worker threads are busy processing other client requests, new requests are forced to wait until a worker thread becomes available. A stack trace can be obtained using the `jstack` command to show the state of the worker threads and the waiting requests.

  > ⓘ **Note:**
  >
  > If all worker threads are processing the same requests for a long time, the server sends an alert that it might be deadlocked. All threads might be tied up processing unindexed searches.

- If a request handler is busy with a client connection, other requests sent through that request handler are forced to wait until it is able to read data. If there is only one request handler, all connections are impacted. Stack traces obtained using the `jstack` command will show that a request handler thread is continuously blocked.
- If the Java virtual machine (JVM) in which the server is running is not properly configured, it can spend too much time performing garbage collection. The effect on the server is similar to that of a network or firewall configuration problem. A stack trace obtained with the `pstack` utility will show that most threads are idle except the one performing garbage collection. It is also likely that a small number of CPUs is 100% busy while all other CPUs are idle. The server will also issue an alert after detecting a long JVM pause that will include details.
- If the JVM in which the server is running has hung, the `pstack` utility should show that one or more threads are blocked and unable to make progress. In such cases, the system CPUs should be mostly idle.
- If a there is a network or firewall configuration problem, communication attempts with the server will fail. A network sniffer will show that packets sent to the system are not receiving TCP acknowledgment.
- If the host system is hung or lost power with a graceful shutdown, the server will be unresponsive.

> ⓘ **Note:**
>
> If it appears that the problem is with the server software or the JVM, work with a support provider to diagnose the problem and potential solutions.

Problems with the administrative console

If you have problems working with the administrative console, there are common reasons why.

If a problem occurs when trying to use the administrative console, reasons might include one of the following:

- The web application container that hosts the console is not running. If an error occurs while trying to start it, consult the logs for the web application container.
- If a problem occurs while trying to authenticate, make sure that the target server is online. If it is, the access log may provide information about the authentication failure.
- If a problem occurs while interacting with the server instance using the administrative console, the access and error logs for that instance can provide additional information.

Troubleshooting problems with SSL communication
Enable TLS debugging and restart the server to save your changes.

Steps

1. Enable TLS debugging in the server to troubleshoot SSL communication issues:

```
$ dsconfig create-debug-target \
  --publisher-name "File-Based Debug Logger" \
  --target-name
com.unboundid.directory.server.extensions.TLSConnectionSecurityProvider \
  --set debug-level:verbose \
  --set include-throwable-cause:true
```

```
$ dsconfig set-log-publisher-prop \
  --publisher-name "File-Based Debug Logger" \
  --set enabled:true \
  --set default-debug-level:disabled
```

2. Make the option take effect on a scheduled server restart.
   a. In the `java.properties` file, add `-Djavax.net.debug=ssl` to the `start-ds` line.
   b. To make the option take effect on a scheduled server restart, run `bin/dsjavaproperties`.

PingDataMetrics Server API reference

The PingDataMetrics Server REST API can be used to build custom dashboards and other applications for processing and viewing data.

The API interface can be accessed using standard tools and charting packages. The PingDataMetrics Server API is also easily accessed from a Web browser.

Connection and security

No sensitive user data is collected by the PingDataMetrics Server and stored in the DBMS. If secure access to the PingDataMetrics Server REST API is required, enable secure HTTPS connections and require authentication.

If not configured during setup, you can enable a secure HTTPS Connection Handler and authentication using `dsconfig`.

> ⓘ **Note:**
>
> By default, the PingDataMetrics Server can open up to 20 simultaneous database connections. The HTTP Connection handler that runs the REST API servlet has a default value of 15 connections. If the PingDataMetrics Server receives requests through multiple HTTP Connection Handlers, make sure that the total number of request handlers does not exceed the maximum number of database connections.

When authentication is enabled, the REST API service requires HTTP basic authentication. Requests are authenticated against entries in the `api-users` LDIF backend, or entries in `cn=Root DNs,cn=config`. Root distinguished name (DN) users have many privileges by default. To restrict access, authenticate with users in the `api-users` backend instead, to prevent the unnecessary use of more privileged account credentials.

Adding a REST API user

Add a REST API user to enable that user to access the API if authentication is enabled.

About this task

Enable REST API authentication by setting the `require-api-authentication` property of the Metrics HTTP Servlet Extension Configuration object as follows:

```
$ bin/dsconfig set-http-servlet-extension-prop \
  --extension-name "PingDataMetrics Server REST API Servlet" \
  --set require-api-authentication:true
```

Perform the following steps to add a REST API user:

Steps

1. Create a file name `api-user1.ldif` containing one or more user entries with no privileges.

   ```
   dn: cn=app-user1,cn=api-users
     changeType: add
     objectClass: inetOrgPerson
     objectClass: person
     objectClass: top
     cn: app-user1
     uid: app-user1
     sn: User1
     userpassword: api1
     ds-pwp-password-policy-dn: cn=Default Password Policy,cn=Password
     Policies,cn=config
   ```

   The password is in clear text. It will be encrypted next.
2. As a privileged user that can add API users, load the entry using the following `ldapmodify` command.

   ```
   $ bin/ldapmodify --filename api-user1.ldif
   ```
3. Authenticate using one of the following options:

   - Authenticate using the full DN of the user added (`cn=app-user1, cn=api-users`).
   - Authenticate using the UID (`app-user1`).

   The user name to DN map is governed by the `identity-mapper` setting of the Metrics REST HTTP Servlet Extension configuration object.
4. Enable Velocity Template authentication with the following command.

   ```
   $ bin/dsconfig set-http-servlet-extension-prop \
     --extension-name Velocity \
     --set require-authentication:true
   ```

Securing error messages

Enable the `omit-error-message-details` Metrics HTTP Servlet Extension Configuration object.

About this task

When developing an application that uses the PingDataMetrics Server API, error messages should not be delivered from the API directly to a user. Also, the application should not depend on error messages or reason text. These messages can change over time, and their presence can depend on server configuration. Use the HTTP return code and the context of the request to create a client error message that displays to the user.

The PingDataMetrics Server API has an `omit-error-message-details` Metrics HTTP Servlet Extension Configuration object. When enabled, this object restricts error messages to the typical reason phrase associated with the HTTP return code, such as `Not Found` for an HTTP `404` error. This prevents the server from inadvertently revealing information about itself or its data.

Steps

- To enable the `omit-error-message-details` object, run the following command:

```
$ bin/dsconfig set-http-servlet-extension-prop \
  --extension-name "PingDataMetrics Server REST API Servlet" \
  --set omit-error-message-details:true
```

Response codes
Response codes tell the user if a request was successful, in error, not found, an internal service error, or the service is not available.

The following response codes are available.

Response code	Description
`200 OK`	The request was processed successfully and the requested data returned.
`400 Bad Request`	The request contained an error. Refer to the error message to resolve the issue.
`404 Not Found`	The requested resource is not found or no samples are collected for the metric.
`500 Internal Server Error`	An unexpected server error occurred. Refer to the error message for more info.
`503 Service Not Available`	The metric query service is temporary offline. Refer to the error message for more info.

The following is a sample response body.

```
<?xml version="1.0" encoding="UTF-8"?>
<errorResponse
 xmlns="com.unboundid.directory.mon.api.v1.models">
<errors reason="unknown_data_source_id" message="There are
  no metrics defined with id connections"/>
</errorResponse>
```

List monitored instances

Get a list of all monitored instances along with their current status.

The default format is JSON. The servlet will use the HTTP Accept header as a hint if no specific format is specified. Results are filtered using the various `instance` query parameters.

URL	`/api/v1/instances`
Method	GET

Formats	JSON, XML
Query parameters	**`instanceHostname`** Hostnames of the servers from which data is gathered. Multiple values are evaluated as logical ORs. **`instanceLocation`** Locations of the servers from which data is gathered. Multiple values are evaluated as logical ORs. **`instanceType`** Types of servers to get data from. Possible values are: ▪ `ds` ▪ `proxy` ▪ `sync` ▪ `metrics-server` **`instanceVersion`** Versions of the servers to get data from. Multiple values are evaluated as logical ORs.

All instances in JSON format.

```
curl \
-X GET \
https://<metricsHost>:8080/api/v1/instances.json
```

All PingDirectory Server and PingDirectoryProxy Server instances in XML format:

```
curl \
-X GET \
https://<metricsHost>:8080/api/v1/instances.xml?
instanceType=data-store&instanceType=proxy
```

Response Code	200 0K

Response Body	{
	"found" : 2
	"offset" : 0,#
	"instances" : [{
	"type" : "ds",
	"id" : "pingidentity4510",
	"hostname":
	"pingidentity5200.example.com",
	"displayName" :
	"pingidentity4510",
	"version": "Directory
	Server 7.3.0",
	"operatingSystem":
	"Linux",
	"status": {

```
        "state": "ONLINE"
        }
   }, {
      "type" : "ds",
     "id" : "unboundid3500",
     "hostname":
 "unboundid3500.example.com",
     "displayName" :
 "directory3500",
     "version": "UnboundID
 Directory Server 3.5.0.0",
     "operatingSystem":
 "Linux",
     "status": {
       "state": "DEGRADED",
       "unavailableAlerts": [
         "replication-
backlogged"
       ]
     }
} ] }
```


Retrieve monitored instance

Get a specific monitored instance along with its status. The default format is JSON. The servlet will use the HTTP Accept header as a hint if no specific format is specified.

URL	`/api/v1/instances`
Method	GET
Formats	JSON, XML
Query parameters	N/A
Server state	The PingDataMetrics Server returns the server state status of the monitored instance, which is displayed by the `status` parameter. The `status` parameter can have one of the following values: **`OFFLINE`** Server cannot be contacted **`STARTING_UP`** Server is starting **`ONLINE`** Server is available **`DEAD_LOCKED`** Server is deadlocked and not able to process more operations **`UNAVAILABLE`** Server is unavailable, but not offline. The server can be in lockdown mode, but still online **`DEGRADED`**

Server is available but is incapable of providing services

CONNECTION_ERROR

Server could not connect or has lost connection to the host

Instance with ID `metrics-server` in JSON format.

```
curl \
-X GET \
https://<metricsServerHost>:8080/api/v1/instances/metrics-
server.json
```

Response Code	200 0K

Response Body	<pre>{
	  "displayName": "metrics-server",
	  "hostname": "metrics-
	server.example.com",
	  "id" : "metrics-server",
	  "operatingSystem": "Linux",
	  "status" : {
	  "state" : "ONLINE"
	  },
	  "type" : "metrics-server",
	  "version": "PingData PingDataMetrics
	 Server 7.3"
	}</pre>

List available metrics

Get a list of metric definitions with their units, dimensions, names, and other values.

The default format is JSON. The servlet will use the HTTP Accept header if no specific format is specified.

URL	`/api/v1/metrics{.format}`
Method	GET
Formats	JSON, XML
Query parameters	**`name`** Limits the results to metrics whose names contain a matching substring. The search is not case-sensitive. **`type`** Limits the results to the metrics of the specified type. Possible values include the following: ▪ `discreteValued` ▪ `continuousValued` ▪ `count`

	group Limits the results to the metrics within the specified group. Possible values include the following: ▪ `PingDirectory Server Backend` ▪ `Monitoring Data Cache` ▪ `Java Virtual Machine` ▪ `LDAP` ▪ `Entry Balancing` ▪ `PingDirectory Server Entry Cache` ▪ `External Server` ▪ `Host System` ▪ `Metric Query` ▪ `Monitoring DBMS` ▪ `Monitoring Data Processing` ▪ `Replication` ▪ `Sync Pipe` **instanceType** Limits the result to metrics that uses the specified instance types as sources. Possible values include the following: ▪ `ds` ▪ `proxy` ▪ `sync` ▪ `metrics-server` **statistic** Limits the results to metrics that provides the specified statistics. Possible values include the following: ▪ `count` ▪ `average` ▪ `maximum` ▪ `minimum` ▪ `histogram`

All metrics in JSON format.

```
curl \
  -X GET \
  https://<metricsServerHost>:8080/api/v1/metrics.json
```

All count type metrics in the “PingDirectory Server Backend” group providing either count or average statistics:

```
curl \
  -X GET \
  https://<metricsServerHost>:8080/api/v1/metrics.json?
type=count&group=ds
```

```
%20backend&statistic=count&statistic=average
```

ⓘ **Note:**

Spaces in parameter values can be encoded as `%20` or `t`.

Response Code	200 0K

Response Body	{
	"found": 7,
	"metrics": [
	{
	"countUnit": {
	"abbreviatedName": "Chkpt",
	"pluralName": "Checkpoints",
	"singularName": "Checkpoint"
	},
	"description": "Number of database
	checkpoints
	performed by the backend",
	"dimensions": [
	{
	"id": "backend",
	"values": [
	"userroot"
	]
	}
	],
	"group": "Directory Server Backend",
	"id": "backend-checkpoints",
	"instanceTypes": [
	"ds"
	],
	"name": "Backend Checkpoints",
	"shortName": "Checkpoints",
	"statistics": [
	"count"
	],
	"type": "count"
	},
	...

Retrieve a metric definition

Use this reference to retrieve a specific metric definition.

The default format will be JSON if none is specified.

ⓘ **Note:**

The servlet will use the HTTP Accept header as a hint if no specific format is specified.

URL	`/api/v1/metrics/{metricId}{.format}`
Method	GET

Formats	JSON, XML
Query parameters	Parameters N/A

Metric with ID `backend-sequential-writes` in XML format.

```
curl \
  -X GET
  https://<metricsServerHost>:8080/api/v1/metrics/backend-
sequential-
writes.xml
```

Response Code	200 0K
Response Body	<?xml version="1.0" encoding="utf-8" standalone="yes"?> <countMetric xmlns="com.unboundid.directory.mon.api.v1" id="backend-sequential-writes" name="Sequential Disk Writes" shortName="Sequential Writes" group="Directory Server Backend"> <description>Number of Sequential I/O Disk writes made by backend</description> <instanceTypes> <instanceType>ds</instanceType> </instanceTypes> <statistics> <statistic>count</statistic> </statistics> <dimensions> <dimension id="backend"> <values> <value>userroot</value> </values> </dimension> </dimensions> <countUnit singularName="Sequential Write" pluralName="Sequential Writes" abbreviatedName="Seq Wr" /> </countMetric>

Perform a metric query

A metric query returns the collected sample data from the various monitored instances. Depending on client requirements, you can present returned data in different ways.

Common query parameters

`instanceType`

Types of instances to get data from. Possible values include the following:

- `ds`

- `proxy`
- `sync`
- `metrics-server`

`instanceLocation`

Location(s) of the instances from which data is collected.

`instanceHostname`

Names of the machines hosting the instances.

`instanceVersion`

Version(s) of the instances providing the data.

`instance`

(multi-valued) – ID(s) of the instances from which data is collected. The instance ID is the `cn` of the external server and the same name as listed by the `status` command.

`startTime`

Include samples on or after the specified time. The time is either an absolute time in ISO 8601 format (such as 2016-08-13T19:36:00Z) or a time relative to the `endTime` (such as -5m or -4h). By default, the start time is -5m.

`endTime`

Include samples on or before this time. The end time is either an absolute time in ISO 8601 format or a time relative to now (such as -5m or -4h). The default end time is now. Offset time values are relative to the current system clock time on the PingDataMetrics Server.

`maxIntervals`

The number of separate intervals, between the start and end times, returned. This is considered the “resolution” of the data over time. By default, the maximum number of intervals is 1, which means all samples collected between the start and end times will be aggregated into one result according to the statistic selected.

`statistic`

Retrieve and apply this statistic to the data. Default for count based metrics is count and average for other metric types. Possible values include the following:

- `count`
- `average`
- `minimum`
- `maximum`
- `histogram`

`dimension`

Include only these dimension values. A colon separates the dimension name and values, which are separated by commas (for example, op-type:add,delete).

`pivot`

Pivot by these dimensions. A pivot keeps the data separated along different dimensional values. The value “instance” can be used to keep the data separate between different instances. For metrics that have the histogram statistic, the histogram pivot can also be used to keep the values of each histogram bucket separate.

`tz`

Specifies the timezone to be used when displaying dates. By default, it is GMT. The timezone is specified in Java Time Zone format. For example, "US/Central" is CST in the United States.

Sub-parameters for the count and average statistics

Both the count and average statistics of count type metrics may have a rate scale applied to occurrences over a period of time using the per sub-parameter. The valid rate scaling values are:

- `s` or `second`
- `m` or `minute`
- `h` or `hour`

Sub-parameters for the histogram statistic

The histogram statistic includes all buckets and keeps the raw value for each bucket. Graphs can be configured to show the percentage of all operations above a given threshold, such as 50 ms. These graphs are useful for looking at a small percentage of operations in a given category. If the value falls between histogram bucket boundaries, the buckets where it falls will be included in the data. Possible values include the following:

`min`

Includes in the calculation only the histogram data above the given threshold.

`max`

Provides an upper bound on the histogram value.

`percent`

Allows the histogram values to be reported as a percentage of the overall values. Instead of returning raw counts, the value is a fraction of the total. This percentage is calculated within a pivot.

ⓘ **Note:**

If both `min` and `max` are specified, the returned value is the sum of all buckets between and including `min` and `max`.

Data set structure

The data set structure is a proprietary data structure that is space-optimized and designed to work with charting libraries like Highcharts or FusionCharts.

The default format for the data set structure is JSON. If no format is specified, the servlet will use the HTTP accept header as a hint.

URL	`/api/v1/metrics/{metricId}/dataset{.format}`
Method	GET
Formats	JSON, XML

ⓘ **Note:**

All of the Common Query parameters apply to this resource.

Get the average response time metric for add and delete operations from 7/7/2015 for all PingDirectory Server and PingDirectoryProxy Server in Austin and Houston:

```
curl \
  -X GET \
  https://<metricsServerHost>:8080/api/v1/metrics/response-time/
dataset?
instanceType=directory-server

 &instanceType=proxy&instanceLocation=austin&instanceLocation=houston&startTime=-1
  &endTime=2015-07-07&pivot=instance&dimension=op-
type:add,delete
```

Get the new connections metric and scale the value per hour in the last 5 minutes:

```
curl \
  -X GET \
  https://<metricsServerHost>:8080/api/v1/metrics/new-
connections/dataset?
statistic=count;per:hour
```

Get the percentage of all occurrences in the last hour where the response-time metric has a value above 50ms:

```
curl \
  -X GET \
  https://<metricsserverHost>:8080/api/v1/metrics/response-time/
dataset?
statistic=histogram;min:50;percent&startTime=-1h
```

Response Code	200 0K
Response Body	When one time interval is requested, a category dataset is returned where the first pivoted dimension values are listed as categories and each data point corresponds to a category. Subsequent pivots and histogram buckets are included as a series and subseries. The following example is the result of two pivots, op-type and instance.


```
{
  "type" : "category",
  "firstSampleTime" : 1344090300000,
  "lastSampleTime" : 1344090600000,
  "metric" : {
  "type" : "discreteValued",
  "id" : "response-time",
  "name" : "Response Time",
  "shortName" : "Response Time",
  "description" : "Time for server
 to process an LDAP
  operation and send a response to
 the client",
  "group" : "LDAP",
  "instanceTypes" : [ "ds",
 "proxy" ],
  "statistics" : [ "average",
 "count", "histogram" ],
```

```
  "dimensions" : [ {
  "id" : "application-name"
  }, {
  "id" : "op-type",
  "values" : [ "Search", "ModifyDN",
 "Add", "Delete",
  "Compare", "Bind", "Modify" ]
  } ],
  "countUnit" : {
  "singularName" : "Operation
 Response Time",
  "pluralName" : "Operation Response
 Time",
  "abbreviatedName" : "Response
 Time"
  },
  "valueUnit" : {
  "singularName" : "Millisecond",
  "pluralName" : "Milliseconds",
  "abbreviatedName" : "Msec"
  }
  },
  "series" : [ {
  "label" : "unboundid35",
  "data" : [ "0", "0", "0", "0",
 "0", "0", "0", "0" ]
  }, {
  "label" : "unboundid3",
  "data" : [ "0", "0", "0", "0",
 "0", "0", "0", "0" ]
  } ],
  "label" : "op-type",
  "categories" : [ "Search",
 "Delete", "Bind", "Modify",
  "Add", "ModifyDN", "Compare" ]
}
```


Google Chart Tools Datasource protocol

Google's Chart Tools Datasource protocol can present metrics data.

The Google Visualization API query language (the tq request parameter) is not supported. The PingDataMetrics Server supports JSON, HTML, CSV, and TSV data formats as outlined by the Datasource protocol.

URL	`/api/v1/metrics/{metricId}/datatable`
Method	GET
Formats	JSON, HTML, CSV, and TSV
Query Parameters	**`tqx=out:html`** HTML formatted output **`tqx=out:csv`** CSV formatted output **`tqx=out:tsv-excel`** TSV formatted output

tz

Specifies the timezone to be used when displaying dates. The Google Visualization API assumes that the times returned are in local time. The PingDataMetrics Server stores and returns all timestamps in GMT. This parameter specifies how the PingDataMetrics Server presents the time. Usually, the client will pass the user's local timezone in IANA Time Zone Database format, such as "US/Central."

ⓘ **Note:**

All Common query parameters apply to this resource.

The following example gets the average response time metric for the last 5 minutes with 30 second (5 * 60 / 10) resolution and pivoted by op-type and then instance in CSV format:

```
curl \
  -X GET \
  https://<metricsServerHost>:8080/api/v1/metrics/response-time/
datatable?
tqx=out:csv&maxIntervals=10
  &pivot=op-type&pivot=instance&tz=US/Central
```

Response Code	200 0K

Response Body	When only one time interval is requested, the first pivoted dimension values form the first column. For queries that request more than one time interval, the start of each time interval forms the first column. Combinations of subsequent pivoted dimension values and/or histogram buckets are included as additional columns. All date and time values are under the GMT time zone.
	`"Time","server35 AVERAGE Milliseconds","server3 AVERAGE Milliseconds"`
	`"2012-08-04T14:38:00Z","0","0"`
	`"2012-08-04T14:39:00Z","0","0"`
	`"2012-08-04T14:40:00Z","0","0"`
	`"2012-08-04T14:41:00Z","0","0"`
	`"2012-08-04T14:42:00Z","0","0"`

The following sample illustrates using Google chart tools.

```
<html>
  <head>
    <!--Load the AJAX API-->
    <script type="text/javascript"
src="https://www.google.com/jsapi"></script>
    <script type="text/javascript">

    // Load the Visualization API and the line chart package.
```

```
  google.load('visualization', '1.0', {'packages':
['corechart']});
    // Set a callback to run when the Google Visualization API
 is loaded.
  google.setOnLoadCallback(drawChart);

function drawChart() {
    var query = new google.visualization.Query
      ('https://<MetricsHost>:8080/
      api/v1/metrics/response-time/datatable?maxIntervals=10
      &pivot=optype&pivot=instance');
    query.send(handleQueryResponse);
   }
  function handleQueryResponse(response) {
    if (response.isError()) {
      alert('Error in query: ' + response.getMessage() + ' '
        + response.getDetailedMessage());
      return;
   }
  var data = response.getDataTable();

  var visualization = new

 google.visualization.LineChart(document.getElementById('chart_div'));
    visualization.draw(data, null);
   }
  </script>
 </head>
 <body>
  <!--Div that will hold the chart-->
  <div id="chart_div"></div>
  </body>
</html>
```

Access alerts

The `eventTypes` and `event` APIs can retrieve information and alerts from monitored servers.

`eventType` API

Provides the range of alert types that have occurred

`events` API

Provides detail about individual alerts.

Retrieve event types

The range of alerts that have been generated by monitored servers can be retrieved, with optional filtering, based on the following API definition.

URL	`/api/v1/eventTypes/[?query-parameters]` retrieves a list of event types.
Method	GET
Formats	JSON, XML

Query Parameters	instance, instanceType, startTime and endTime. For a description of each parameter, see *Perform a metric query*.

Response Code	200 0K
Response Body	`["health-check-available-to-degraded","health-check-degraded toavailable"]`

Retrieve events

The detailed information for one or more events can be retrieved, with optional filtering, based on the following API definition.

URL	`/api/v1/events/[?query-parameters]` retrieves a list of events. `/api/v1/events/{eventId}` retrieves a single event.
Method	GET
Formats	JSON, XML
Query Parameters	**type** Limits the result to include only events of the specified types. See the HTML API Reference for event types. **severity** Limits the result to include only events that have the matching severity. Valid severity values are: INFO, WARNING, ERROR, and FATAL. **instance, instanceType, instanceLocation, instanceHostname, instanceVersion, startTime, and endTime.** For a description of each parameter, see *Perform a metric query*. **limit, offset** For a description of each parameter, see *Pagination*.

Response Code	200 0K
Response Body	`{ "found" : 2, "offset" : 0, "events" : [`

```
 {"id":"9bdfd1b8-3811-4a84-
b779-93553ff35f83",
 "creationDate":1351274815559,
 "eventType":"server-starting",
 "eventSeverity":"INFO",
 "sourceProductInstance":"lockdown-
test",
 "summary":"Server Starting",
 "detail":"The Directory Server is
 starting"},
 {"id":"9bdfd1b8-3811-4a84-
b779-93553ff35f83",
 "creationDate":1351274815559,
 "eventType":"server-starting",
 "eventSeverity":"INFO",
 "sourceProductInstance":"directory-3",
 "summary":"Server Starting",
 "detail":"The Directory Server is
 starting"}
  ]
}
```


LDAP SLA

The LDAP service level agreement (SLA) API lists the LDAP SLA objects (configuration data) and queries any single LDAP SLA object.

The query of an LDAP SLA object results in the aggregated LDAP SLA configuration, scalar data containing current values for the LDAP SLA, and time-series data. Current data comes from the Threshold object. Historical data comes from a metric query.

Historical data is more expensive to fetch and is only included if the client requests it. This allows an LDAP SLA query to efficiently get the configuration and current data for clients that only need the current data. A client that needs both current and historical data can include the appropriate query parameter and get all the data in a single call.

Retrieve the SLA object

List the LDAP service level agreement (SLA) objects (configuration data) and query any single LDAP SLA object.

The default format will be JSON if none is specified. The servlet will use the HTTP Accept header as a hint if no specific format is specified.

URL	`/api/v1/sla/ldap` returns a list of all LDAP SLA configuration objects in name-order. This includes current values and status as held by the Threshold objects, but will only include any historical data. `/api/v1/sla/ldap/{sla-name}` returns a single LDAP SLA configuration object plus optional historical data.
Method	GET
Formats	JSON, XML
Query Parameters	For the first URL: **`instance`**

	Returns LDAP SLA's that reference the specified instance. **application-name** Returns LDAP SLA's that reference this application name. **ldap-op** Returns LDAP SLA's that reference this LDAP operation. For the second URL: **historical** multi-valued, optional: ▪ `time` - Includes time series data. ▪ `limits` - Includes the percent of time thresholds limits that have been exceeded. Requires Threshold. ▪ `alerts` - Includes all Threshold alerts. Requires thresholding. ▪ `histogram` - includes response-time histogram as column data. ▪ `nines` - Includes response time values that correlate to 99%, 99.9%, 99.99%, and 99.999% response-time measurements) **startTime** (optional). The time at which the historical data starts. The default is `1hr`. **endTime** (optional). The time at which the historical data ends. The default is `5m`. **pivot** (optional) Historical time-series pivots by this dimension ▪ `instance` - pivot by producing server. ▪ `ldap-op` - pivot by LDAP operation. ▪ `histogram` - pivot response-time series by histogram buckets. **maxIntervals** (optional). Number of points to include in the historical time series. The default is 100.

Retrieving an SLA object.

```
curl -X GET http://x3550-
09:8080/api/v1/sla/ldap/Acme+Identity+Portal?historical=time
```

```
\&historical=nines\&pivot=instance\&startTime=-15m
```

Response Code	200 0K

Response Body	{
	"name":"Acme Identity Portal",
	"applicationName":"Application 5",
	"ldapOps":["search"],
	"servers":
	["x2270-08.pingidentity.lab:1389"],
	"enabled":true,
	"responeTimeState":"NORMAL",
	"throughputState":"normal",
	"currentResponseTime":6.002752,
	"currentThroughput":7032.794,
	"averageResponseTime":6.212055,
	"averageThroughput":5517.1323,
	"responseTimeWarnLimit":8.0,
	"responseTimeCriticalLimit":10.0,
	"throughputWarnLimit":8000.0,
	"throughputCriticalLimit":10000.0,
	"responseTimeSeries":{
	"type":"timeInterval",
	"firstSampleTime":1359045070000,
	"lastSampleTime":1359045970000,
	"rateScaling":"NONE",
	"statistic":"AVERAGE",
	"metric":{
	"type":"discreteValued",
	"id":"response-time",
	"name":"Response Time",
	"shortName":"Response Time",
	"description":"Time for server to
	process
	an LDAP operation and send a
	response to the client.",
	"group":"LDAP",
	"instanceTypes":["directory-
	server","proxy"],
	"statistics":
	["average","count","histogram"],
	"dimensions":[{"id":"application-
	name",
	"values":["unidentified directory
	application",
	"unidentified proxy
	application","application 9",
	"application 5","root user","admin
	user",
	"application 6"]},{"id":"op-
	type","values"
	
	["search","modifydn","add","delete","compare",
	"bind","modify"]}],
	"countUnit":
	{"singularName":"Operation Response
	Time",
	"pluralName":"Operation Response
	Time",
	"abbreviatedName":"Response
	Time"},

	<pre> "valueUnit": {"singularName":"Millisecond", "pluralName":"Milliseconds","abbreviatedName":" }, ...</pre>

Pagination

Pagination is supported for both the metrics and instances listing URLs.

Query parameters

limit

Specifies the maximum number of results to return. The default is to return all results.

offset

Specifies how many results to skip for the first results to return.

Response parameters

found

The number of results that satisfied the query parameters.

offset

The index into the total result set where the current response begins.

PingDirectory Security Guide

Introduction

The PingDirectory Security Guide provides concepts and procedures to secure and manage the PingDirectory platform.

Storing and handling consumer identity data requires taking appropriate steps to safeguard it while continuing to provide fast, real-time, and available services for the consumers who consent to its use.

This guide is intended for administrators responsible for installing and managing servers in an enterprise or consumer-facing identity environment.

Administrators should already know the following:

- Identity platforms and LDAP concepts
- General system administration and networking practices
- Java Virtual Machine (JVM) compromise
- Application performance monitoring

Threat vectors in an identity environment

The PingDirectory platform serves as the authentication repository for a wide variety of network applications, and it often stores sensitive user information and application data.

Hackers that obtain user credentials can cause extensive damage to individuals, systems, and businesses.

Business costs to secure and monitor identity deployments can be large, but the total cost is small compared to the cost of a security breach. A security breach requires resources to investigate the incident, assess the scope of the damage, identify any compromised data, and revert any changes. Affected users must be notified and must be compensated for downtime and for any costs incurred from the exposure of their personal data. However, the damage to a company's reputation is often the most costly result.

Directories are the central component within identity management systems. They streamline authentication and authorization across system boundaries. Whether for user, account, or subscriber provisioning, directory services must be properly secured so that sensitive information is not accessible by unauthorized individuals externally or internally. If the directory service is compromised, attackers can gain access to the data that it contains and to other systems that rely on the directory service for authentication and authorization.

In securing the directory service, several threat vectors must be considered:

- Compromise of:
  - The underlying host system on which the PingDirectory Server is running
  - Peripheral systems that might have access to server data, including backups, LDIF exports, log files, or monitor data
  - Systems used in setting up new instances including orchestration frameworks and configuration-as-code repositories
- Access to network communication

  This includes not only the ability to observe traffic passing between clients and servers, but also the potential to intercept and alter that communication, or impersonate a legitimate server.
- Attacks that involve communication with the server over the intended protocols, including LDAP, SCIM, and the Directory REST API, whether by unauthenticated clients, regular and administrative users using legitimately obtained credentials, or attackers who might be able to obtain compromised credentials

  This includes denial of service attacks and attempts to access unauthorized data.

Securing the host system

The process of securing the underlying systems used to run the PingDirectory Server or that might have access to server data is outside the scope of this document but is covered in other security references. However, it is important to rely on a number of best practices, as outlined in the following sections.

Minimize installed software

Each application or command on a system is potentially a security hole that could provide unauthorized users a way to get into the system.

The PingDirectory software has a minimal set of dependencies. As a pure Java application, its primary dependency is the Java Virtual Machine (JVM). However, troubleshooting and support data collection tools might rely on other operating system utilities to obtain information about the state of the underlying system. On Linux, these tools include:

- **`cat`**
- **`crontab`**
- **`df`**
- **`dmesg`**
- **`dstat`**
- **`ethtool`**
- **`gstack`**
- **`ifconfig`**
- **`iostat`**
- **`ip`**

- **jinfo**
- **jmap**
- **journalctl**
- **jps**
- **jstack**
- **ls**
- **mpstat**
- **netstat**
- **pidstat**
- **pmap**
- **ps**
- **pstack**
- **sar**
- **sh**
- **ss**
- **sysctl**
- **systemctl**
- **tail**
- **top**
- **tuned-adm**
- **udevadm**
- **uname**
- **uptime**
- **vmstat**

Consider removing software that is not a critical part of the operating system, such as not needed to run the JVM, and that is not needed to run the previous troubleshooting and data collection tools.

Keep systems patched

Keep any software that cannot be removed from the system up to date.

Install all operating system security patches and software updates in a timely manner.

Operating system and software updates do have the potential to cause unforeseen problems. Thoroughly test all updates in a staging environment prior to production.

Monitor security-related mailing lists and news feeds, and consider following security experts on social media. The sooner you know about operating system or software vulnerabilities, the sooner you can take action to minimize the exposure while waiting for and testing the appropriate patches.

Minimize network services

Take steps to reduce the potential for compromise of network services.

Steps include:

- Disable any unnecessary network services.
- If there are network daemons that must run on the system but are only accessed over the loopback interface, such as a local SMTP server for relaying email messages, configure them so that they are not accessible to external clients.
- Use firewall software to ensure that only the minimum number of ports are exposed to external systems.
- When possible, configure services to run as a non-root user with as few rights as possible.

Configure filesystem security

The most basic forms of filesystem protection are file permissions and filesystem encryption.

Any portion of the filesystem that contains sensitive data should be accessible only to the account used to run the server. In a default PingDirectory software installation, all components of the server reside within the instance root. When the software is extracted, which should be done using the account that will be used to run the software, the instance root directory will have filesystem permissions of 0700, preventing any access by other accounts on the system other than those exempt from file permission restrictions, like the root account. Directories used to hold database files are also given permissions of 0700 by default, and log files are written with default permissions of 0600. If the server is configured to access other areas of the filesystem outside of the instance root, take care to set file permissions and ownership on the paths that contain that content.

Some operating systems offer mechanisms beyond the basic file permissions. For example, Linux systems offer **`getfacl`** and **`setfacl`** commands that can define more fine-grained access controls for files and directories. Consider using those mechanisms to provide greater protection from unauthorized access.

Filesystem auditing software can help identify questionable use of file permissions. It can also keep a record of all filesystem permission and content changes. Although this is not useful for content that changes frequently like database and log files, it can be very helpful for detecting changes to other content, like server binaries and configuration. We also recommend using this auditing for the operating system binaries and configuration.

PingDirectory software provides support for encrypting database contents, backups, LDIF exports, and other content. You can also gain additional protection by enabling filesystem encryption to help protect against unauthorized access to the underlying storage.

> ⓘ **Note:**
>
> The use of filesystem encryption might not offer much additional protection for a mounted filesystem because it appears unencrypted to the users and applications that interact with it.

Enable time synchronization

Always set the system clock to the right time.

There are several reasons for this, including:

- It ensures that logging and auditing timestamps are accurate.
- It is useful when correlating events across multiple systems.
- It is essential to ensure correct behavior if replication conflicts arise.
- It is necessary for time-sensitive authentication mechanisms like GSSAPI and UNBOUNDID-TOTP.
- It is important for time-related password policy processing.

Apply recommended OS-level tuning

The PingDirectory administration guide makes a number of tuning recommendations to help ensure software runs smoothly.

These include:

- Configure file descriptor limits.
- Configure process limits.
- Apply recommended filesystem tuning.
- Disable filesystem swapping.
- Configure filesystem event monitoring limits.
- Tune the I/O scheduler.
- Tune memory management.
- Manage OS-level environment variables.

- Configure the system to allow the PingDirectory software to listen on privileged ports while running as a non-root user.

Run the PingDirectory software in a container

Consider running the PingDirectory software in a Docker image or some other type of container.

This can help isolate the server from other software and processes on the system. It also encourages adopting a DevOps philosophy, which streamlines the process for deploying new instances of the software using a reliable and repeatable process. This is good for disaster recovery because it allows a new instance to be quickly created if an existing instance should fail for any reason.

Containerization can also offer other security-related benefits. For example, containers frequently do not provide direct shell access to the instance, reducing the chance that an attacker can gain access to server data through that avenue. If you subscribe to the DevOps blue-green strategy, in which configuration changes are applied by spinning up new instances with the desired settings rather than updating existing instances, unauthorized configuration changes are easier to detect and revert.

Maintain the Java Virtual Machine

Keep the Java Virtual Machine (JVM) up to date with the latest patches.

Run the most recent major release of the JVM that the PingDirectory software supports because it can offer support for additional security features that are not available in older versions.

In some cases, you might want to run the PingDirectory software on a standalone JVM installation rather than one provided through the operating system's package management system. This might not be necessary when using the DevOps configuration-as-code practice because the process of updating the JVM or applying operating system patches typically involves spinning up a new container using the updated software. However, if you use a more traditional deployment model in which the software is installed on long-lived systems that are updated over time, then relying on a JVM provided by the operating system vendor might cause it to be updated unexpectedly. By maintaining a dedicated Java installation for the PingDirectory software, you have better control over when the JVM is updated and the specific version that is in use.

Maintain a support contract with your JVM vendor to ensure that they will be able to assist with any issues that you encounter in the Java runtime itself.

Minimize access to the underlying system

If administrators can access the underlying system, then the mechanism they use to do so should be as secure as possible.

All shell access must occur over an encrypted session that uses strong authentication, such as relying on SSH keys rather than passwords, and ideally using a second authentication factor like TOTP or U2F.

Keep the number of users authorized to access the system to a minimum, and ensure each authorized user has their own account rather than sharing accounts across multiple individuals. This makes it easier to audit access and identify which administrator made a given change. Using separate accounts also avoids problems that might arise whenever the credentials need to be changed. With a separate account per administrator, each has their own distinct credentials.

The account used to run and manage the PingDirectory software itself should not be able to directly authenticate to the underlying system. Instead, each user authorized to interact with the PingDirectory software on the underlying system must authenticate with their own account before using some mechanism, such as su or sudo on UNIX-based systems, to interact with the server software.

The accounts for the users who manage the PingDirectory software and the account used to run that software should be configured with the minimum set of capabilities required to perform their duties. Those who manage the underlying system might require full access, but those who just maintain the PingDirectory software can be given restricted access. If possible, restrict the set of commands that those users can invoke and their access to other running processes and areas of the filesystem.

If a system account becomes compromised or its owner leaves, terminate that account as quickly as possible. While managing system accounts is usually best left to a centralized naming service like LDAP, this is not recommended for the accounts used to manage the PingDirectory software itself because an issue with the server might prevent users from authenticating to the system to address it. Local file-based accounts are the most reliable, but it is important to keep a current list of all users with access to these systems and of all systems they can access with those credentials so that the credentials can quickly be revoked if necessary.

Managing the server without shell access to the underlying system
Even if PingDirectory Server administrators are not granted shell access to the underlying system, it is still possible to manage the server.

Most administrative functions can be performed remotely over secure LDAP or HTTP connections.

The web-based administration console provides support for managing the server configuration and schema. It also provides access to a variety of status information, including monitor entries, active alarms, and administrative alerts.

If you extract the PingDirectory software onto your local system, then you will also have access to a variety of command-line tools that can interact with the server remotely. Some of the most useful tools include:

`status`

Retrieve a variety of status information from the server.

`dsconfig`

Manage the server configuration.

`dsreplication`

Manage and monitor replication.

`collect-support-data`

Collect a wide variety of information that is useful for troubleshooting problems and understanding the server configuration and status. The resulting support data archive can be securely streamed back to the client system.

`backup`

Back up the contents of one or more server backends. The backup files will be written onto the server filesystem.

`restore`

Restore a backup stored on the server filesystem.

`export-ldif`

Export the contents of a specified backend to LDIF. The LDIF file will be written onto the server filesystem.

`import-ldif`

Import LDIF data stored on the server filesystem into a specified backend.

`config-diff`

Compares server configurations, whether of two different servers or different versions of the configuration from the same instance, to identify differences.

`ldapsearch`

Search for information stored in the server.

`ldapmodify`

Update information stored in the server, including creating new entries or updating or removing existing entries.

`ldappasswordmodify`

Reset user passwords.

`manage-account`

Manage password policy state for users.

`ldap-diff`

Compare the data between multiple servers to identify differences.

`audit-data-security`

Examine and report on various security-related aspects of data stored in the server.

`schedule-exec-task`

Schedule an administrative task that can be used to execute a specified command on the server system. This task is not enabled by default, and it provides a number of safeguards to ensure that it cannot be invoked by unauthorized users and that authorized users are not allowed to invoke unauthorized commands.

You might also need to access files on the server filesystem, especially for things like backups, LDIF exports, and log files. There are options for this that do not require shell access:

- Consider using a secure shared filesystem that is accessible from other trusted systems. Even if you don't want to place the server root itself on a shared filesystem, you could write backups, LDIF exports, and rotated log files to it so that they are more readily available.
- Use the file servlet that is provided as part of the PingDirectory Server installation. If you go to https://*server-address*:*server-https-port*/instance-root/ and authenticate as a user with the **`file-servlet-access`** privilege, which is included in the default set of root privileges, you can see a listing of all files and directories in the server instance root and you can download any files of interest to your desktop.

Use system logging and auditing

Auditing provides vital information for diagnosing problems or investigating security breaches.

Most operating systems provide an audit mechanism that records information about system events. This can include basic information like keeping a record of sign on attempts, but it might also be possible to capture more detailed information like recording each command that is invoked or each file that is accessed.

Make sure that logging and auditing are properly tuned to record an appropriate amount of information without impeding system performance. You might also want to ensure that system logs are recorded locally and sent to a remote system to ensure higher availability and to reduce the likelihood that an attacker who gains access to the system will be able to cover their tracks.

Configuring data encryption

While attackers ideally won't be able to gain access to any of the systems running the PingDirectory software or to systems that contain data generated by the PingDirectory software, you should still take additional measures to protect the data.

One of the best ways to accomplish this is to encrypt all sensitive data. The PingDirectory software provides extensive support for data encryption. It is possible to encrypt the backend databases, both entry contents and index data, as well as the replication database and the LDAP-accessible changelog if enabled. It is also easy to encrypt backups, LDIF exports, log files, and support data archives.

Enabling data encryption during setup

Data encryption should be enabled when running setup, which ensures that all data added to the server is encrypted and also configures the server to automatically encrypt backups and LDIF exports.

The interactive setup process, which is started when setup is run without any arguments, guides you through the process of enabling data encryption, but if you're using non-interactive setup or manage-profile setup, then data encryption can be enabled by providing one of the following arguments.

Argument	Description
`--encryptDataWithPassphraseFromFile`	Specifies the path to a file that contains the passphrase to use to generate the encryption settings definition that encrypt the data. If you provide the same passphrase when setting up multiple instances of the server, then each generates the same encryption settings definition, and each instance can access data encrypted by the other instances.
`--encryptDataWithSettingsImportedFromFile`	Specifies the path to a file that contains one or more encryption settings definitions to be imported into the newly created encryption settings database. Use the `--encryptionSettingsExportPassphraseFile` argument to provide the path to a file containing the passphrase used to encrypt those definitions. If you import the same encryption settings definitions into all servers in the topology, then each instance can access data encrypted by the other instances. See the Exporting encryption settings definitions section for more information on exporting the contents of the encryption settings database.
`--encryptDataWithRandomPassphrase`	Indicates that the server should enable data encryption with an encryption settings definition created from a randomly generated passphrase. If you use this option to set up multiple instances, then they will not have the same encryption settings definitions, and data encrypted by one instance is not accessible on other instances unless the encryption settings definitions are synchronized across all of those instances.

Managing the encryption settings database

If data encryption is enabled in the PingDirectory software, the server use a secret key to actually perform that encryption. Without that encryption key, the encrypted data is worthless.

The PingDirectory software uses an encryption settings database to manage the keys that it uses for data encryption. Each encryption settings definition includes not only the key used to perform the encryption, but also specifies the cipher transformation that is used for that encryption. The encryption settings database can include multiple definitions, but only one of those definitions is marked as the preferred definition, and that is the one that is used for encrypting new data that is written to the database.

The PingDirectory Server provides an `encryption-settings` tool that can manage the contents of the encryption settings database.

Listing encryption settings definitions
Use the **`encryption-settings list`** command to obtain a list of the definitions in the server's encryption settings database.

This command does not require any arguments.

```
$ bin/encryption-settings list
Encryption Settings Definition ID:
 125480AFA3300CD8E48CDF53AF3A0D4DDC8760E115C5B986530F1FE438B19DBD
     Alternate ID:  3D21C37492A2AE8D28BBB7A39A238987005AA4DC
     Description:  Generated during setup on 'test' using a
 passphrase read from a file
     Create Time:  Wed Aug 26 15:04:12 CDT 2020
     Cipher Transformation:  AES/CBC/PKCS5Padding
     Key Length (bits):  128
     Preferred for New Encryption:  true

Encryption Settings Definition ID:
 CA8A76C13DD5CC3F85A437119D9DC0867396910F64E228962A30FF80B36C3B63
     Alternate ID:  7AF6A51673D0BC10915B23370448C1592905E8E2
     Description:  Generated during setup on 'test' using a
 passphrase read from a file
     Create Time:  Wed Aug 26 15:04:12 CDT 2020
     Cipher Transformation:  AES/CBC/PKCS5Padding
     Key Length (bits):  256
     Preferred for New Encryption:  false
```

Creating encryption settings definitions
To create a new encryption settings definition, use the **`encryption-settings create`** command, which takes the following arguments.

Argument	Description
`--cipher-algorithm`	A required argument that specifies the base cipher that should be used for encryption and decryption. The value for this argument must be the name of a valid symmetric encryption algorithm that is supported by the Java Virtual Machine (JVM). We recommend using AES.
`--cipher-transformation`	An optional argument that allows you to specify the complete cipher transformation for encryption performed using this definition. If provided, the cipher transformation should consist of three components separated by forward slashes: the cipher algorithm, the cipher mode, and the padding algorithm, such as "AES/CBC/PKCS5Padding". If this is not provided, the server automatically selects the transformation based on the specified cipher algorithm.
`--key-length-bits`	The length, in bits, to use for the encryption key. When using the AES cipher, this should generally be either 128 or 256.

Argument	Description
`--prompt-for-passphrase`	An optional argument that indicates that the tool should interactively prompt for the passphrase to use when generating the encryption key.
`--passphrase-file-path`	An optional argument that provides the path to a file containing the passphrase that should be used when generating the encryption key.
`--description`	An optional argument that can be used to provide a human-readable description for the new encryption settings definition.
`--set-preferred`	An optional argument that indicates that the new definition should be the preferred definition used for subsequent encryption operations.

The following is an example of the command with some of the arguments included.

```
$ bin/encryption-settings create \
     --cipher-algorithm AES \
     --cipher-transformation AES/GCM/PKCS5Padding \
     --key-length-bits 256 \
     --prompt-for-passphrase \
     --description "An example encryption settings definition"
Enter the encryption passphrase:
Confirm the encryption passphrase:

Successfully created a new encryption settings definition with ID
494DCE52DE58D0A44E56B9E80FC62B257870F2FC7CEEDCA150F4EF51829D7B20.
```

Each encryption settings definition has an underlying passphrase that is used to generate the encryption key. If you provide the `--prompt-for-passphrase` argument, then the tool interactively prompts you for that passphrase. If you provide the `--passphrase-file-path` argument, then it reads the passphrase from a file. If you do not provide either argument, then the tool generates a strong random passphrase for use by that definition.

We generally recommend that you explicitly specify the passphrase for new encryption settings definitions rather than allowing the tool to generate a random passphrase. If you want to create the encryption settings definition across multiple instances, then providing the same passphrase for each instance ensures that the same definition is created everywhere. Further, if you need to decrypt a file that was encrypted with a key from the encryption settings database, like a backup, LDIF export, or log file, then you can decrypt the file (using the **`encrypt-file`** tool or using the PassphraseEncryptedInputStream class provided in the UnboundID LDAP SDK for Java) if you know the passphrase, even if you're on a system that doesn't have access to the server's encryption settings database.

Removing encryption settings definitions

To remove an encryption settings definition, use the **`encryption-settings delete`** command.

This command takes the following arguments.

Argument	Description
`--id`	A required argument that indicates which encryption settings definition should be removed.

Argument	Description
`--no-prompt`	An optional argument that indicates that the specified definition should be removed without prompting for confirmation.

The following is an example of the command with one of the arguments included.

```
$ bin/encryption-settings delete \
     --id 494DCE52DE58D0A44E56B9E80FC62B257870F2FC7CEEDCA150F4EF51829D7B20
If the encryption definition is being used to encrypt existing objects in
 the system,
then those objects will no longer be able to be decrypted. In certain cases,
 the
server may not be able to be restarted until those objects are deleted. Are
 you sure
about deleting it (yes/no)?  yes
Successfully deleted encryption settings definition
494DCE52DE58D0A44E56B9E80FC62B257870F2FC7CEEDCA150F4EF51829D7B20.
```

Before you delete an encryption settings definition, you should make sure that it is not still in use. If you remove an encryption settings definition that is still in use within the database, then any data encrypted with that key becomes inaccessible and operations that attempt to access it fail. See *Re-encrypting data in the database* for more information about this.

Also note that the tool will not allow you to remove the preferred encryption settings definition. If you want to remove the preferred definition, then you must make another definition the preferred definition. See the *Setting the preferred encryption settings* definition for more information.

Exporting encryption settings definitions

To use the same set of encryption settings definitions, ensure that all servers in the topology are configured.

There are two ways to accomplish this:

- Use the **`encryption-settings create`** command on each instance with the same passphrase. Alternatively, if you're enabling data encryption when running setup, provide the same passphrase file to each instance.
- Create the desired definitions on one instance, export them from that instance, and import them into the other instances or provide the export file when running setup.

To export one or more encryption settings definitions, use the **`encryption-settings export`** command, which supports the following arguments.

Argument	Description
`--output-file`	A required argument that specifies the path to the export file to be written.
`--pin-file`	An optional argument that specifies the path to a file containing the passphrase to use to encrypt the contents of the export. If this is not provided, the tool interactively prompts for the passphrase. Because this passphrase is used to protect the contents of the export, it must be strong and it should not match the passphrase used to create any of the definitions.

Argument	Description
`--id`	An optional argument that can be used to explicitly specify the IDs of the definitions to include in the export. If this is not provided, then all definitions are included.
`--use-legacy-export-format`	Indicates that the tool should use a legacy export format that was supported by older versions of the server. You might need to use this argument if you are exporting definitions from a newer version for import into an older version. The legacy export format can only hold a single encryption settings definition, so the `--id` argument must be used to specify the ID of the definition to include.

The following is an example of the command with one of the arguments included.

```
$ bin/encryption-settings export \
     --output-file exported-definitions.esd
Enter the PIN to use to encrypt the definition:
Re-enter the encryption PIN:
Successfully exported encryption settings data to file exported-
definitions.esd.
```

Importing encryption settings definitions
To import definitions into the encryption settings database, use the **`encryption-settings import`** command.

It supports the following arguments:

Argument	Description
`--input-file`	A required argument that specifies the path to the file containing the exported encryption settings definitions.
`--pin-file`	An optional argument that specifies the path to a file containing the passphrase used to protect the contents of the encryption settings definitions. If this is not provided, then the tool interactively prompts for the passphrase.
`--set-preferred`	An optional argument that indicates that if the export file contains a definition that is marked as preferred, then it becomes the new preferred definition for the instance.

The following is an example of the command with one of the arguments included.

```
$ bin/encryption-settings import \
     --input-file exported-definitions.esd
Enter the PIN used to encrypt the definition:
Successfully imported encryption settings definition
125480AFA3300CD8E48CDF53AF3A0D4DDC8760E115C5B986530F1FE438B19DBD from file
/ds/PingDirectory/exported-definitions.esd.
Successfully imported encryption settings definition
```

```
CA8A76C13DD5CC3F85A437119D9DC0867396910F64E228962A30FF80B36C3B63 from file
/ds/PingDirectory/exported-definitions.esd.
```

When importing definitions into a non-empty encryption settings database, the existing definitions are preserved and the new definitions are added alongside them. If any of the definitions from the import file already exist in the database, they are skipped when processing the import.

Setting the preferred encryption settings definition
To choose a different definition to be the one that is preferred for new encryption operations, use the **`encryption-settings set-preferred`** command.

This command supports the following arguments.

Argument	Description
`--id`	A required argument that specifies the ID of the encryption settings definition that should become the new preferred definition.

The following is an example of the command with one of the arguments included.

```
$ bin/encryption-settings set-preferred \
     --id CA8A76C13DD5CC3F85A437119D9DC0867396910F64E228962A30FF80B36C3B63
Encryption settings definition
CA8A76C13DD5CC3F85A437119D9DC0867396910F64E228962A30FF80B36C3B63 was
 successfully set
as the preferred definition for subsequent encryption operations.
```

When creating a new definition that is used across multiple instances, you might want to omit the `--set-preferred` argument when running `encryption-settings create`. Instead, you should ensure that the definition is created across all instances first, and then use `encryption-settings set-preferred` to make it the new preferred definition. This helps avoid the chance that an instance that has not yet been updated with the new definition encounters an error when trying to decrypt data from another instance that is already using that definition.

Re-encrypting data in the database
If you update the server to use a new preferred encryption settings definition, that new definition is used for all subsequent data encryption, but existing data remains encrypted with an older key.

In many cases, this can be acceptable. Records in the replication database or the LDAP changelog that were encrypted with an old key are eventually removed in accordance with the configured purge settings, and entries in backends are upgraded to use the new key as they are updated by clients.

However, there can be cases, such as if you suspect that an existing key might have been compromised, in which you want to immediately transition to using a new definition. To do this, you must use the encryption-settings create command to add the desired new encryption settings definition to the settings database on all servers or alternatively, create the new definition on one server, export it from that instance, and import it into all of the other instances. You can then use the encryption-settings set-preferred command to make the new definition the preferred definition across all instances. Then, complete the following process for each instance, one instance at a time, in the topology:

1. To disable replication for all backends and remove the server from the replication topology, use **`dsreplication disable`**.
2. Stop the instance so that its data will not change during the process of converting to the new key.
3. To export the contents of all encrypted backends, those backed by the Berkeley DB Java Edition database, to LDIF, use the **`export-ldif`** command. If the LDAP changelog is enabled, then also export its contents to LDIF using the same process. Make sure that the LDIF exports are encrypted (see Encrypting LDIF exports) to keep them secure.
4. To re-import the LDIF data into the appropriate backends, use the **`import-ldif`** command.

5. Remove the `changelgoDb` directory in the server instance root, which holds the replication database not the `db/changelog` directory, which holds the database for the LDAP changelog.
6. Start the server.
7. To add the server back into the replication topology and verify that changes are properly replicated between that instance and other servers in the topology, use **`dsreplication enable`**.

Managing data encryption in the global configuration

If data encryption is not enabled during setup, you can enable it at any time by ensuring that the server is configured with an appropriate encryption settings definition and updating the following properties in the global configuration.

Property	Description
`encrypt-data`	Indicates whether data encryption should be enabled. Upon enablement, any writes to backends, the replication database, and the LDAP changelog are encrypted, but existing data remains unencrypted. Any unencrypted data in the replication database and LDAP changelog is eventually removed in accordance to their purging configuration, but we recommend exporting backends to LDIF and re-importing to ensure that all of the data that they contain is encrypted.
`encryption-settings-cipher-stream-provider`	The cipher stream provider that should be used to protect the contents of the encryption settings database. See the *Configuring cipher stream providers* topic for more detail.
`encrypt-backups-by-default`	Indicates whether any new backups that are created should automatically be encrypted with a key from the encryption settings database. If you want to create a backup that is not encrypted, then you can provide the `--doNotEncrypt` argument to the backup command. If you want to create a backup that is encrypted with a different key, then use one of the `--promptForEncryptionPassphrase`, `--encryptionPassphraseFile`, or `--encryptionSettingsDefinitionID` arguments.
`backup-encryption-settings-definition-id`	The ID of the encryption settings definition that is used when encrypting backups by default. If this is not specified, then the server's preferred encryption settings definition is used.

Property	Description
`encrypt-ldif-exports-by-default`	Indicates whether any new LDIF exports that are created sho485uld be automatically encrypted with a key from the encryption settings database. As with the backup tool, the export-ldif tool offers the `--doNotEncrypt`, `--promptForEncryptionPassphrase`, `--encryptionPassphraseFile`, and `--encryptionSettingsDefinitionID` arguments to change its encryption behavior.
`ldif-export-encryption-settings-definition-id`	The ID of the encryption settings definition that is used when encrypting LDIF exports by default. If this is not specified, then the server's preferred encryption settings definition is used.
`automatically-compress-encrypted-ldif-exports`	Indicates whether the server should automatically compress LDIF exports that are encrypted.

Configuring cipher stream providers

Cipher stream providers are used to protect the keys stored in the encryption settings database.

By default, setup generates a strong, random passphrase that it writes to a file, and the server uses a file-based cipher stream provider to read the passphrase from that file and use it to generate a key used to encrypt the contents of the encryption settings database. However, the server supports additional cipher stream providers that can use alternative means for unlocking the encryption settings database. Options include:

- Require a passphrase to be interactively provided when the server is started, or any time an external process needs access to the encryption settings database.
- Use a key stored in the Amazon Key Management Service (KMS).
- Use a key stored in a HashiCorp Vault instance

It is also possible to use the Server SDK to create cipher stream providers that use custom logic to protect the contents of the encryption settings database.

If you want to configure the server to use a different cipher stream provider, first ensure that the desired cipher stream provider is defined and enabled in the configuration and then update the global configuration to use that cipher stream provider to protect the encryption settings database. You should do this with the server online so that it can automatically re-encrypt the encryption settings database with the new key.

For example, to configure the server to use the Amazon KMS cipher stream provider, first define the cipher stream provider as appropriate in the server configuration using a change as in the following example.

```
dsconfig create-cipher-stream-provider \
     --provider-name "Amazon KMS" \
     --type amazon-key-management-service \
     --set enabled:true \
     --set "aws-access-key-id:[KMS_ACCESS_KEY_ID]" \
     --set "aws-secret-access-key:[KMS_SECRET_ACCESS_KEY]" \
     --set "kms-encryption-key-arn:[KMS_KEY_ARN]"
```

Then, update the global configuration to use the new cipher stream provider.

```
dsconfig set-global-configuration-prop \
     --set "encryption-settings-cipher-stream-provider:Amazon KMS"
```

See the `use-the-amazon-kms-cipher-stream-provider.dsconfig` and `use-the-vault-cipher-stream-provider.dsconfig` batch files in the `config/sample-dsconfig-batch-files` directory for more information about the KMS and Vault cipher stream providers.

Encrypting backups

Even if the data stored in a backend's database is encrypted, there is additional benefit in encrypting backups of that database.

The encryption covers additional database metadata that isn't encrypted, and it also serves as a kind of integrity check to ensure that the backup hasn't been altered or corrupted since it was created.

If you enable data encryption when running setup, then the server is automatically configured to encrypt backups by default. If encryption is enabled after setup, you can use the `encrypt-backups-by-default global` configuration property to configure this. In either case, the default behavior is to use the preferred encryption settings definition to obtain the encryption key, but you can explicitly specify an alternative definition for backups using the backup-encryption-settings-definition-id property.

The backup tool offers the following arguments related to encryption.

Argument	Description
`--encrypt`	Indicates that the backup should be encrypted. This can be used to explicitly enable encryption if the `encrypt-backups-by-default global` configuration property is set to false. This argument is also required if you use one of the `--promptForEncryptionPassphrase`, `--encryptionPassphraseFile`, or `--encryptionSettingsDefinitionID` arguments. If encryption is not enabled by default in the global configuration and this argument is provided without one of the `--promptForEncryptionPassphrase`, `--encryptionPassphraseFile`, or `--encryptionSettingsDefinitionID` arguments, then the server's preferred encryption settings definition is used.
`--promptForEncryptionPassphrase`	Indicates that the backup tool should interactively prompt for the passphrase used to generate the encryption key. If this is provided, then the backup is encrypted with that key rather than one obtained from an encryption settings definition.
`--encryptionPassphraseFile`	Specifies the path to a file that contains the passphrase that should be used to generate the encryption key. If this is provided, then the backup is encrypted with that key rather than one obtained from an encryption settings definition.
`--encryptionSettingsDefinitionID`	Specifies the identifier for the encryption settings definition that should be used to encrypt the data. This can override the logic that the server would otherwise use to select the encryption settings definition.

Argument	Description
`--doNotEncrypt`	Indicates that the backup should not be encrypted. This can be used to explicitly obtain an unencrypted backup if `encrypt-backups-by-default` is set to true in the global configuration.

Each backup directory includes a descriptor file with information about all of the backups contained in that directory. This descriptor indicates whether the backup is encrypted, and if it was encrypted with a definition from the encryption settings database, then it includes its ID. In such cases, the restore tool automatically obtains the necessary key from the encryption settings database.

However, if the backup was encrypted with a passphrase rather than an encryption settings definition or if the definition is not included in the encryption settings database but you know the passphrase used to create that definition, then you can use one of the following arguments to provide the necessary passphrase.

Argument	Description
`--promptForEncryptionPassphrase`	Indicates that the restore tool should interactively prompt for the passphrase used to generate the encryption key.
`--encryptionPassphraseFile`	Indicates that the restore tool should interactively prompt for the passphrase used to generate the encryption key.

Encrypting LDIF exports

Each backup is essentially an archive that includes all of the necessary database files. It can be quickly restored in a PingDirectory Server if the need arises, but the data is not accessible for any other purpose.

LDIF is a standard plain-text representation of LDAP data. While it might take somewhat longer to import data from an LDIF file than it does to restore a backup, the PingDirectory Server can generally import entries at a very high rate. Because LDIF is a standard format, the data can be easily parsed with a wide variety of libraries (including the UnboundID LDAP SDK for Java) or viewed in a text editor. It is also highly compressible, and a gzipped LDIF file can take up substantially less space than a backup (which includes additional data beyond the entries, like indexes and database structure). As such, there are compelling reasons to use LDIF exports as a backup mechanism instead of or in addition to backup archives. However, because it is a plain-text format, it is essential that LDIF exports be encrypted to prevent their content from being readily available to anyone who gains access to them.

The `encrypt-ldif-exports-by-default` property in the global configuration can automatically encrypt LDIF exports by default. The `ldif-export-encryption-settings-definition-id` property can specify the encryption settings definition to protect the data, or the server can fall back to using the preferred definition. Further, the `automatically-compress-encrypted-ldif-exports` property can be used to compress the data as it is encrypted for substantial space savings.

If the global configuration is not set up to encrypt LDIF exports by default, or if you want to encrypt the data with a different key, the **`export-ldif`** tool provides the following arguments which are largely the same as those offered by the backup tool:

- `--encryptLDIF`
- `--promptForEncryptionPassphrase`
- `--encryptionPassphraseFile`
- `--encryptionSettingsDefinitionID`
- `--doNotEncrypt`

When importing an LDIF file, the server can automatically determine whether the data is encrypted or compressed, and if the file was encrypted with a key from the encryption settings database, the encryption header includes the ID of the definition that was used. However, if the LDIF data was encrypted with a passphrase rather than an encryption settings definition, then one of the following arguments can be used to provide that passphrase:

- `--promptForEncryptionPassphrase`
- `--encryptionPassphraseFile`

Encrypting, sanitizing, and signing log files

Log files are useful for understanding server usage patterns and troubleshooting problems, but they can also contain sensitive data like attribute values that might appear in entry DNs or search filters. To protect against this, the server can log to encrypted files.

File-based loggers include the following configuration properties to control this.

Property	Description
`encrypt-log`	Indicates whether the log file should be encrypted.
`encryption-settings-definition-id`	Specifies the ID of the encryption settings definition that should be used to obtain the encryption key. If this is not specified, then the preferred definition is used.

If you need to access data in an encrypted log file, then the **`encrypt-file`** tool can be used to decrypt its content. This tool is discussed in more detail in the encrypt-file tool section. However, it might not be necessary to decrypt log files to be able to use them. Both the **`search-logs`** and **`summarize-access-log`** tools both provide support for operating on encrypted and compressed log files.

In most cases, no special handling is needed, because the log data is encrypted with a definition from the server's encryption settings database, and the tool can obtain the appropriate definition from that database. However, if the encryption settings database is not available, such as if the tool is run from a system other than the one on which the server is running, or no longer contains the definition that was used to encrypt the log file, then the `--encryptionPassphraseFile` argument can be used to specify the passphrase used to generate that definition.

For additional information, see the `config/sample-dsconfig-batch-files/create-encrypted-loggers.dsconfig` batch file.

Sanitizing log files

Another way to prevent unauthorized access to sensitive information in log files is to remove or obscure that information.

The **`sanitize-log`** tool can be used to accomplish this. It classifies each log field into one of three categories.

Category	Description
Preserve	The value of the field is preserved as it appeared in the log message. The **`sanitize-log`** tool is preconfigured with a set of log fields that should not contain any sensitive information and are considered safe to preserve, but you can add additional fields to this set using the `--preserveField` argument.

Category	Description
Tokenize	The value of the field is converted into a token, which is a number surrounded by curly braces (for example, the first tokenized value is “{1}”, the second is “{2}”, and so on). If the field value appears to be a DN or search filter, then only attribute values in that DN or filter are tokenized; otherwise, the entire value is tokenized. The same token is used for the same value every time it appears in a log file, which can make it easier to correlate information across operations without revealing what the value actually is. The tool is preconfigured with a set of log fields that are appropriate for tokenization, but you can add additional fields to this set with the `--tokenizeField` argument.
Redact	The entire value of the field will be replaced with the string `---REDACTED---`. Any field that is not marked for preservation or tokenization is automatically redacted. If you want to redact a field whose value would otherwise be preserved or tokenzied by default, you can use the **`--redactField`** argument.

The **`sanitize-log`** tool automatically detects whether the log file is encrypted or compressed, and you can also optionally encrypt or compress the output. It provides the following arguments in support of this.

Argument	Description
`--inputEncryptionPassphraseFile`	Specifies the path to a file containing the passphrase needed to decrypt the contents of the log file. This is generally not needed, as log files are encrypted with a key from the encryption settings database and the **`sanitize-log`** tool can automatically obtain the appropriate key from that database. However, if that key is not available for some reason, you can use this argument to provide the necessary passphrase.
`--compressOutput`	Indicates that the sanitized output should be compressed.
`--encryptOutput`	Indicates that the sanitized output should be encrypted.
`--outputEncryptionPassphraseFile`	Specifies the path to a file containing the passphrase that is used to encrypt the sanitized output. If this argument is not provided but the `--encryptOutput` argument is given, then the tool interactively prompts for the passphrase.

Signing log files
Regardless of whether they are encrypted, the server can digitally sign log files to provide a means of verifying that the content has not been altered in any way. This can be controlled by the `sign-log` property in the configuration for each logger.

Rather than signing log files as a whole, the server signs groups of one or more messages. Each time it writes a set of log messages to disk, a signature is generated for that set of messages. In the event that log messages are altered, or a set of messages are removed from the file, this provides a more fine-grained method for determining which content is trustworthy and which is not. Signature information can also carry over between rotated log files, so it is possible to determine if an entire log file has been removed.

The **`validate-file-signature`** tool can be used to verify the signatures in a log file to confirm that the content has not been altered. This tool supports the following arguments.

Argument	Description
`--file`	Specifies the path to the file whose signature should be validated. If a chain of log files should be validated, then this should be the most recent file in the chain.
`--encryptionPassphraseFile`	Specifies the path to a file containing the passphrase that was used to encrypt the file contents. This should not be necessary if the file was encrypted with a key from the encryption settings database and that key is still accessible. If this argument is not provided and the encryption passphrase cannot be automatically retrieved, then the tool interactively prompts for the passphrase.
`--validateLogChain`	Indicates that the tool should validate a chain of log files. It starts with the file specified by the `--file` argument, but if that file was created after rotating from a previous file, then it works its way backwards through the chain of log files. ⓘ **Note:** When the server is restarted, it cannot continue using the same signature chain that it was using before the restart, so the process of validating a chain automatically stops when it encounters a server restart.
`--numFiles`	Specifies the maximum number of log files in the chain that should be validated. By default, the tool attempts to verify as much of the chain as possible.
`--logDuration`	Specifies the minimum length of the time span that should be covered by the log content when validating a chain of files. If this is specified, then its value should be given as an integer followed by a time unit (for example, “10 minutes” or “1 day”), and the tool tries to iterate backwards through files in the chain until at least this length of time has been covered.

Argument	Description
`--ignoreMultipleSignedBlocks`	Indicates that the tool should ignore errors that can arise if a log file contains multiple signed blocks. This can happen if the server was restarted and the logger is configured to append to any existing log file rather than rotating it and starting with a fresh log file.
`--ignoreMissingEndOfSignature`	Indicates that the tool should ignore an error if the target log file does not end with valid signature information. This might happen when trying to validate the active log file with the server still online.
`--ignoreMissingFile`	Indicates that the tool should ignore an error caused by attempting to follow a log file chain when a file indicates that it was created after rotating from an earlier log file, but that earlier log file does not exist. This might happen if the older log file has been deleted by log retention processing.

Encrypting TOTP secrets and delivered tokens

The server might sometimes need to store authentication-related data in the user's entry in reversible form.

This includes:

- Shared secrets that are needed to generate time-based passwords for use by the UNBOUNDID-TOTP Simple Authentication and Security Layer (SASL) mechanism. It might also need to store the last TOTP password that the client used to prevent it from being reused.
- One-time passwords that have been generated by the server and delivered to the user for use by the UNBOUNDID-DELIVERED-OTP SASL mechanism.
- Password reset tokens that can be used to allow a user to recover access to their account even if they have forgotten their password, if their password is expired, or if the account has been locked.

This information needs to be stored in the server in a form that allows the server to obtain its clear-text value. Although these values can actually be stored in the clear, the **Encrypt TOTP Secrets and Delivered Tokens** plugin can be used to encrypt their values so that they are not available to anyone who gains access to the corresponding operational attributes in the user's entry.

For more information, see the `config/sample-dsconfig-batch-files/enable-encryption-for-shared-secrets-and-one-time-passwords.dsconfig` sample batch file.

Encrypting support data archives

The **`collect-support-data`** tool is a vital resource for support personnel to use when trying to diagnose issues with a PingDirectory Server.

It collects a range of information about the server installation and the underlying system, including:

- The current server configuration and configuration changes made over time
- The server schema
- Portions of log files
- Server monitor data
- The server root DSE
- The output of the status command
- A list of all of the access control rules defined in the server
- Information about all third-party extensions that have been installed

- Stack traces of all threads running in the server
- Information about JVM garbage collection and memory usage
- A listing of all processes running on the system
- Information about the underlying performance of the system, including CPU utilization, memory consumption, disk I/O
- Other system-related information, including networking and storage configuration

The **collect-support-data** tool attempts to redact sensitive information (like encoded passwords and other credentials) as it's collecting data, and you can use the `--securityLevel` argument to configure how aggressive it is when deciding what to obscure or remove. However, even at the highest security level, the resulting support data archive likely has information that you want to protect.

One way that you can do this is by encrypting the support data archive before you provide it to support personnel. You can encrypt the contents with a passphrase, and then provide the passphrase to the support engineer through a different channel than was used to transmit the archive. The **collect-support-data** tool offers the following arguments related to encrypting the archive.

Argument	Description
`--encrypt`	Indicates that the support data archive file should be encrypted.
`--passphraseFile`	Specifies the path to a file containing the passphrase to use to encrypt or decrypt the file. If this argument is not provided, then the tool interactively prompts for the passphrase.
`--generatePassphrase`	Indicates that the tool should generate a random encryption passphrase and write it to the specified passphrase file. This argument should only be used in conjunction with both the `--encrypt` and `--passphraseFile` arguments.
`--decrypt`	Indicates that the tool should decrypt the support data archive at the specified path.

Other files that can be encrypted

The PingDirectory Server and accompanying tools support interacting with a variety of other types of encrypted files.

Examples of this include:

- The files containing the PIN needed to access a certificate key or trust store, such as the `ads-truststore.pin`, `keystore.pin`, and `truststore.pin` files in the server's config directory, can be encrypted.
- If a command-line tool needs to read a password from a file, such as when using the `--bindPasswordFile`, `--keyStorePasswordFile`, or `--trustStorePasswordFile` arguments offered by LDAP-enabled tools, it should be able to read from encrypted files.
- If a command-line tool supports obtaining default argument values from a properties file, such as from `config/tools.properties`, that properties file can be encrypted.
- When writing its output to one or more files, the **ldapsearch** tool can encrypt the data as it is written.
- When reading the set of changes to process, the **ldapmodify** and **parallel-update** tools can read those changes from encrypted LDIF files.
- LDIF tools like **ldifsearch**, **ldifmodify**, and **ldif-diff** support reading from and writing to encrypted LDIF files.

The encrypt-file tool

The PingDirectory Server includes a command-line tool that can be used to encrypt and decrypt files using keys from the server's encryption settings database or using a passphrase that you provide interactively or in a file.

This tool offers the following arguments.

Argument	Description
`--input-file`	Specifies the path to the file containing the data to be encrypted or decrypted. If this is not provided, then the data can be read from standard input, such as entered interactively or piped from another command.
`--output-file`	Specifies the path to the file to which the encrypted or decrypted data is written. If this is not provided, the data is written to standard output.
`--decrypt`	Indicates that input is expected to be encrypted, and the tool should decrypt it. If this argument is not provided, then the tool encrypts the input data.
`--prompt-for-passphrase`	Indicates that the tool should interactively prompt for the passphrase to use to encrypt or decrypt the input data. If this is provided, then the `--input-file` argument must also be provided because the tool does not support both prompting for a passphrase and reading the data to process from standard input.
`--passphrase-file`	Specifies the path to a file containing the passphrase to use to encrypt or decrypt the input data.
`--encryption-settings-id`	Specifies the identifier for an encryption settings definition that is used to encrypt or decrypt the input data.
`--use-topology-key`	Indicates that the data should be encrypted or decrypted using a key that is generated by and shared among servers in the replication topology. This is a legacy encryption mechanism that is no longer used by modern versions of the server, and it is only needed when encrypting data that might need to be decrypted by older instances in the same topology.
`--compress-output`	Indicates that the output should be gzip-compressed as it is written. When the tool is operating in encrypt mode, the data is compressed before it is encrypted.
`--decompress-input`	Indicates that the input data is gzip-compressed. When operating in decrypt mode, the data is decompressed after it has been decrypted.

Argument	Description
`max-megabytes-per-second`	The maximum rate at which the tool should write the encrypted or decrypted data. This can be helpful when operating on large files as a way of avoiding excessive disk I/O that might impact the performance of other I/O operations on a busy server.

The `--prompt-for-passphrase`, `--passphrase-file`, `--encryption-settings-id`, and `--use-topology-key` arguments are all mutually exclusive and cannot be used together. If none of these arguments is provided, then the tool uses a key from the encryption settings database. When encrypting data, it uses the preferred definition. When decrypting data that was encrypted with an encryption settings definition, the encryption header at the beginning of the file should contain the identifier for the appropriate definition.

Centralized logging

By default, the PingDirectory Server logs to files on the server filesystem. This provides the best option for performance and reliability of the logging mechanism itself.

However, this also has the following disadvantages:

- If the system crashes in an unrecoverable manner, such as if the storage becomes corrupted, then the log data might be lost.
- If an attacker is able to gain access to the underlying system, they might be able to alter or delete log files to cover their tracks. Even if the log files are signed so that you can tell that log files have been modified, that doesn't help you determine what the original content was.
- It requires more effort to analyze log files and aggregate results when they are spread across multiple systems.
- In some cases, such as when running in a container like Docker, it might not be easy or possible to get direct access to the instance filesystem.

These issues can be addressed by centralizing log content, and PingDirectory software offers several options to assist with this.

> ⓘ **Note:**
>
> Because PingDirectory Server allows you to define multiple loggers of the same type, you can both log to local files and to one or more centralized locations. This can provide the best combination of usefulness and availability.

Logging to a shared filesystem

One simple option for centralizing log content is to have the server log to a shared filesystem.

This is the simplest option, but it only addresses some of the issues outlined previously. While a shared filesystem reduces the need for access to the instance filesystem, it does not necessarily help prevent attackers from altering the files. If attackers gain access to the underlying system and that system has access to files from other instances, they might be able to cause greater disruption than if the content had been kept local to each instance.

Copying files to a centralized system

Another option to centralize logging is to continue to write log files to the local filesystem, but to periodically copy them to a centralized system.

For greater security, the centralized system can pull the files from the instances rather than having the instances push the content, which avoids allowing the server instances to write data to the centralized system.

However, this option still has some security risk associated with it. If an attacker is able to alter log files, then those altered versions will be copied to the centralized system. This can be mitigated to an extent by copying the content more frequently and using versioning when the same copy is copied multiple times, but there is still a window of time in which an attacker can alter the file before it is copied.

Ingesting logs into a log management system

Many organizations use a centralized log management system, such as Splunk or DataDog. In these cases, product log messages can be ingested into that system to make them accessible from a common location, and to provide improved support for analytics and taking other actions upon log content.

There are several ways that log content can make it into the log management system. Some of these options include:

- Most log management software provides agent software that can read data from log files and send it to the centralized system. In some cases, the agent can stream log data as it is written, which reduces the chance that an attacker has a chance to alter it. In other cases, it can copy the data at configured intervals.
- For cases in which the target application is running in a container like Docker, a common practice is to have that application write log messages to standard output or standard error, and to forward those streams to the log management software. To help support this, the PingDirectory Server provides an option to write access and error log messages, with each message formatted as a JSON object for greater parsability, to standard output or standard error. For more information about logging to standard output or standard error, see the `config/sample-dsconfig-batch-files/enable-console-based-logging.dsconfig` batch file.
- The log management software can provide an API that applications can use to write log messages directly to that service. Although the PingDirectory software does not provide support for this out of the box, it is possible to use the Server SDK to write a custom log publisher to take advantage of this API.

Logging with syslog

The PingDirectory Server can write log messages using the syslog protocol, for both access and error logs.

This allows messages to be aggregated at the system level and potentially forwarded to a centralized system. The messages are written to syslog as they are generated, so attackers do not have a chance to alter these log messages.

If you want to use syslog-based logging, configure the server to log to a syslog server running on the local server over the loopback interface. The local syslog server can then forward the messages to a remote server over a secure connection.

Logging over TCP for improved reliability is now supported.

UDP-based communication is in the clear, so a network observer can see all of the log messages. Therefore, we recommend only using syslog to log to a local syslog server and having it forward messages to a remote server in a secure manner. TLS encryption for TCP-based communication is now optionally supported, so you can safely configure the server to log directly to a remote syslog server.

UDP does not provide any feedback in regard towhether messages are successfully delivered, but TCP does provide this feedback. When using TCP-based logging, you can optionally specify information about multiple syslog servers; if the primary syslog server becomes unavailable, the logger can fail over to an alternative syslog server.

Logging access and error log messages can be logged as JSON objects or in legacy space-delimited text format.

In addition to access and error logging over syslog, loggers that can write JSON-formatted audit and HTTP operation log messages are also provided.

Logging to a remote database

The PingDirectory Server can write log messages to a relational database.

This should work with any database that provides a driver that allows it to integrate with the Java Database Connectivity (JDBC) framework. As with syslog, these log messages are written to the database as they are generated in the server, so attackers do not have a chance to alter them before they are written.

Communication with the database can be secured using whatever facilities are provided by the JDBC driver. In most cases, this includes support for TLS encryption.

Custom loggers created with the Server SDK

The UnboundID Server SDK provides support for creating custom access, error, and HTTP operation loggers.

If necessary, you can create a custom logger that handles these log messages in whatever way you want.

TLS overview

TLS is a popular protocol for securing network communication.

It is the successor to the SSL, and those terms are sometimes used interchangeably. For legacy compatibility purposes, the PingDirectory Server and client tools often use the term SSL in reference to TLS.

TLS can sit below other network protocols in the communication stack to allow those protocols to communicate in a secure manner. The PingDirectory server supports TLS to secure communication with many types of systems, but clients most often use it in conjunction with LDAP, where LDAP secured with TLS is often referred to as LDAPS, and HTTP, referred to as HTTPS.

The security that TLS provides comes in the form of two main components: encryption and trust. It provides a way for two systems to communicate over a secure channel that cannot be deciphered or altered by observers, and it also provides mechanisms for clients to have assurance that they are actually communicating with the intended system.

TLS relies on certificates. This section provides a baseline understanding of certificates, the TLS protocol, the manage-certificates tool, and configuring and using TLS encryption in conjunction with the PingDirectory software.

Understanding X.509 certificates

Although other types of certificates exist, X.509 certificates are the most common type and the only type that the PingDirectory Server supports.

The current version of the specification is X.509v3, which is described in *RFC 5280*. An X.509v3 certificate includes the following components:

- The X.509 encoding version, which makes it possible to differentiate between an X.509v3 certificate or one that conforms to an earlier or future version of the specification.
- The certificate's serial number, which is an integer value that uniquely identifies a certificate as issued by a certification authority. While they were often generated in sequential order in the past, they are now required to be unpredictable with at least 20 bits of entropy.
- A subject DN, which is the distinguished name for the certificate that often provides information about the context in which the certificate is to be used. Certificate subject DNs are discussed in more detail later.

- An issuer DN, which is the distinguished name for the issuer certificate (that is, the certificate used to sign the certificate). For a self-signed certificate, this matches the subject DN.
- A validity window, which indicates the time frame that the certificate should be considered valid. This includes a “notBefore” element, which is the earliest time that the certificate should be considered valid, and a “notAfter” element, which is the latest time that the certificate should be considered valid.
- A public key, which is the public portion of a pair of two cryptographically linked keys. Certificate key pairs are discussed in more detail later.
- An optional subject unique ID, which is intended to uniquely identify the certificate. This has been deprecated in favor of the subject key identifier extension and is generally omitted from X.509v3 certificates.
- An optional issuer unique ID, which is the subject unique ID of the issuer certificate (if it had one). This has been deprecated in favor of the authority key identifier extension.
- An optional set of extensions that provide additional context for the certificate and how it can be used. Certificate extensions are discussed in more detail later.
- A signature, which is a type of cryptographic proof that the certificate really did come from the issuer and has not been altered in any way. As its name implies, a self-signed certificate is signed with its own private key; otherwise, it is signed with the issuer’s private key.

Certificate subject DNs

A certificate’s subject distinguished name (DN) is a name that provides information about the certificate and how it is intended to be used.

Like an LDAP DN, it is comprised of a comma-delimited series of attribute-value pairs, but the attribute names in a certificate subject DN are typically written in all uppercase, whereas they are typically lowercase or camelCase in LDAP DNs.

Attributes that commonly appear in certificate subjects include:

CN

A common name. For a listener certificate, this is often a hostname that clients use to access the certificate, although the subject alternative name extension provides a better mechanism for accomplishing that. Most certificate subject DNs include at least the CN attribute.

E

An email address.

OU

An organizational unit (department) name.

O

An organization (company) name.

L

A locality (city) name.

ST

A state or province name.

> ⓘ **Note:**
>
> This should be the full name of the state or province, not an abbreviation.

C

An ISO 3166 country code (not the full country name).

A certificate subject should include at least one attribute-value pair, and the CN attribute is typically present. Other attributes can be omitted, but the O and C attributes are also fairly common. For example, a listener certificate for a server with an address of ldap.example.com run by the US-based company Example Corp might have a subject of `CN=ldap.example.com,O=Example Corp,C=US`.

Certificate key pairs
Each certificate has a key pair, which consists of two keys that are cryptographically linked so that if you encrypt data with one of those keys, then it can only be decrypted with the other key.

While it is a relatively simple mechanism to come up with a key pair when generating both keys at the same time, it is extremely difficult (in cryptographic terms, computationally infeasible) to derive one key from the other.

When generating a key pair, one of these keys is designated the public key, and the other is designated the private key. The public key can be made widely available, but the private key should be kept secret and not shared with anyone. As long as that is the case, then you can use this key pair to perform two different functions.

Function	Description
Encryption (also called confidentiality)	If someone wants to send you a secret message and doesn't want anyone else to be able to read it, they can encrypt it with your public key, and since you are the only one with the private key, only you can decrypt it.
Digital signatures	If you encrypt data with your private key, then it can only be decrypted with your public key. Since your public key can be made widely available, then this encryption doesn't actually protect the content, but it does prove that the message came from you because only your private key could have generated it. ⓘ **Note:** When generating a digital signature, you typically don't encrypt the entire message, but rather a hash of the message such as using a digest algorithm like SHA-256. This can also provide integrity protection because if the decrypted signature matches the digest of the original message, then it guarantees that not only the message came from you, but that it hasn't been altered since you signed it.

There are two primary public key algorithms that are used in certificates used for TLS communication: RSA and EC. RSA is based on multiplying really big prime numbers together, while EC is based on computations involving special types of elliptic curves.

RSA is more widely supported, but it's slower and requires bigger keys to achieve the same level of security as EC. If you need to support legacy clients, then you probably need to use an RSA certificate, and you should choose a key size of at least 2048 bits. But if all of your clients support elliptic curve certificates, then EC might be the better choice, with a key size of at least 256 bits.

Certificate extensions
Extensions provide additional context for a certificate.

There are several types of extensions, but some of the most common include.

Extension	Description
Subject key identifier	Holds a unique identifier for the certificate, which is generally derived from the certificate's public key.
Authority key identifier	Holds the subject key identifier for the issuer certificate. It can help identify the issuer certificate, especially when presented with an incomplete certificate chain.
Subject alternative name	Holds a list of ways that clients are expected to reference a server when establishing a connection to it. Clients should take this information into account when deciding whether to trust a server's certificate. There are several types of values, but the most common are DNS names, IP addresses, and URIs. ⓘ **Note:** DNS names should be fully qualified, but can optionally use an asterisk in the leftmost component to match any single name in that component, For example, "*.example.com" could match "www.example.com" or "ldap.example.com", but would not match "ldap.east.example.com" or "example.com".

Extension	Description
Key usage	Provides information about the way in which the certificate is expected to be used. Allowed key usages include: **digitalSignature** Indicates that the certificate can be used for digitally signing data, excluding certificates and CRLs. **nonRepudiation (also known as contentCommitment)** Indicates that the certificate can be used to prevent denying the authenticity of a message. **keyEncipherment** Indicates that the certificate can be used to protect encryption keys, such as symmetric keys derived during TLS key agreement. **keyAgreement** Indicates that the certificate's public key can be used for key agreement, such as deriving the symmetric key used to protect TLS communication. **keyCertSign** Indicates that the certificate can be used for signing other certificates. For example, it can act as a certification authority. **cRLSign** Indicates that the certificate can be used to sign certificate revocation lists (CRLs). **encipherOnly** When used in conjunction with the keyEncipherment usage, this indicates that the public key can only be used for encrypting data during key agreement. **decipherOnly** When used in conjunction with the keyEncipherment usage, this indicates that the public key can only be used for decrypting data during key agreement

Extension	Description
Extended key usage	Acts as an alternative to the key usage extension and provides additional high-level functionality. Allowed extended key usages include: **serverAuth** Indicates that the server might present the certificate to the client during TLS negotiation. **clientAuth** Indicates that the client might present the certificate to the server during TLS negotiation. **codeSigning** Indicates that the certificate can be used to sign source and compiled code. **emailProtection** Indicates that the certificate can be used to sign or encrypt email messages. **timeStamping** Indicates that the certificate can be used to assert the time that an event occurred. **ocspSigning** Indicates that the certificate can be used to sign an OCSP (online certificate status protocol) response.
Basic constraints	Indicates whether the certificate can act as a certification authority and, if so, the maximum number of intermediate certificates that might appear beneath it in a certificate chain.

Certificate chains

A certificate chain is an ordered list of one or more certificates, in which each subsequent certificate is the issuer of the previous certificate.

During TLS negotiation, the server presents a certificate chain to the client so that the client can determine whether it should trust that chain and continue with the negotiation. The client can optionally present its own certificate chain to the server.

If a certificate is self-signed, then its chain contains only that one certificate. If a certificate was signed by a root certificate authority (CA), that is, a self-signed certification authority certificate, then the chain contains two certificates: the server certificate followed by the CA certificate. If there is a single intermediate CA, then the chain contains the server certificate, followed by the intermediate CA, and finally the root CA, and so on.

Intermediate certification authorities are useful security purposes, especially for commercial CAs or other widely trusted authorities. If a client trusts a root CA certificate, then it likely trusts anything that has that root CA certificate at the base of its chain. This means that it’s critical to keep the root CA certificate secure, because if it is compromised, then any certificate signed by it (whether directly or indirectly) can no longer be trusted. With intermediate CA certificates, the root certificate can be kept offline in very secure storage and only used whenever it is necessary to sign a new intermediate CA certificate. The intermediate CA certificates can be the ones to sign end-entity certificates. They should also be extremely well-guarded

because it would be disastrous if one of them were compromised requiring it and all certificates it signed had to be revoked, but at least the root CA could be used to sign a new intermediate CA certificate, and that could then be used to sign replacements for all certificates that had been issued by the former CA certificate.

> ⓘ **Note:**
>
> The certificate chain that the server presents to the client (or that the client presents to the server) during TLS negotiation does not necessarily have to be the complete chain.

If the root CA at the end of the chain is widely trusted, then the server might assume that the client already has that root CA in its default set of trusted certificates and leave it off the chain with the assumption that the client will retrieve it from its default trust store. While the same could theoretically be true for intermediate CA certificates, only the root CA certificate is commonly omitted. Upon receiving such an incomplete chain, the client should look in its default trust store to see if it contains the issuer certificate which it can identify using things like the issuer DN or an authority key identifier extension.

The certificate at the head of a certificate chain (the first one in the list) is often called the end-entity certificate. If it's at the head of a chain presented by a server during TLS negotiation, then it can also be called the server certificate, whereas if it's at the head of a chain presented by a client, then it can also be called a client certificate. The certificate at the tail of the complete chain must be a root CA certificate. In the case of a self-signed certificate, the chain contains only a single certificate that serves both of those roles.

Representing certificates, private keys, and certificate signing requests
X.509 is an encoding format that uses the ASN.1 distinguished encoding rules (DER), which is a binary format. When writing a certificate to a file, it can use this raw DER format, or it can use a plain-text format called PEM.

The PEM encoding consists of a line containing the text `-----BEGIN CERTIFICATE-----`, followed by a set of lines containing the base64-encoded representation of the raw DER bytes (typically with no more than 64 characters per line), followed by a line containing the text `-----END CERTIFICATE-----`.

The X.509 encoding contains a certificate's public key, but not its private key. The encoding for private keys is described in the PKCS #8 specification in *RFC 5958*. This also uses a DER encoding, with a PEM variant that uses `-----BEGIN PRIVATE KEY-----` and `-----END PRIVATE KEY-----`, rather than `-----BEGIN CERTIFICATE-----` and `-----END CERTIFICATE-----`. RFC 5958 also describes an encrypted representation of the private key although PingDirectory tools do not currently support that format.

The certificate signing request (CSR) format is described in the PKCS #10 specification in *RFC 2986*. It uses a DER encoding with a PEM variant. The PEM variant uses a header of `-----BEGIN CERTIFICATE REQUEST-----` and a footer of `-----END CERTIFICATE REQUEST-----` although some implementations use the alternate, nonstandard forms `-----BEGIN NEW CERTIFICATE REQUEST-----` and `-----END NEW CERTIFICATE REQUEST-----`.

Understanding certificate trust

Whenever the server presents its certificate chain to the client during TLS negotiation, the client needs to decide whether it wants to trust that certificate chain and believes that it's actually talking to the legitimate server and not an impostor.

If a client is tricked into communicating with a rogue application instead of the real server, such as through DNS hijacking, then that application could steal the client's credentials or trick the client into believing that it has performed some action that it really hasn't. If the rogue application forwards client communication on to the legitimate server, then it could be even harder for the client to detect the trickery, and even easier for that application to steal data or alter communication. This is why it's vital to avoid using "trust all" or "blind trust" options in a production environment.

The steps that a client should take when determining whether to trust a server certificate chain include:

- The client should ensure that it has the complete certificate chain. If the server presented an incomplete chain, then the client should ensure that it can complete the chain with information in an explicitly provided trust store or a default trust store. If the client can't complete the certificate chain, then it should not be trusted.
- The client should ensure that each subsequent certificate in the chain is the issuer certificate for, and that its private key was used to sign, the certificate that precedes it. If a certificate chain contains extraneous certificates, or if a subsequent certificate was not the issuer for the certificate that precedes it, in which case the certificate's signature would not match the expected value, then the chain should not be trusted.
- The client should ensure that it has a reason to trust the certificate at the root of the chain. This is generally done by ensuring that the root certificate authority (CA) certificate can be found in either a trust store configured for use by that client or a default trust store provided by the operating system, Java Virtual Machine (JVM), browser software, or other trusted source. If the client does not have any prior knowledge of the root CA certificate, then the chain should not be trusted.
- The client should verify that the current time is within the validity window for each certificate in the chain. If any certificate in the chain has a notBefore value that is later than the current time or a notAfter value that is earlier than the current time, then the chain should not be trusted.
- The client should verify that the server certificate, the one at the head of the chain, is suitable for the server that the client thinks it's communicating with. It should verify that the address it used to establish the connection matches an address contained in the certificate's subject alternative name extension or in the CN attribute of the certificate's subject.

> ⓘ **Note:**
>
> The client can perform additional types of validation. For example, if any of the root or intermediate certification authorities maintains a certificate revocation list (CRL) or supports the online certificate status protocol (OCSP), then the client should verify that none of the certificates in the chain has been revoked. It can verify that the CA certificates include the basic constraints extension and that the server certificate isn't too many levels deep. Other checks, like those using certificate policy extensions, can also be performed.

Understanding key and trust stores

A key store, or keystore, is a type of database for holding certificates.

Key stores come in multiple forms, but the most common include:

- A file using the Java-specific Java Key Store (JKS) format.
- A file using the standard PKCS #12 format, as described in *RFC 7292*.
- A collection of files that hold certificates and private keys, typically in PEM or DER format.
- A hardware security module (HSM) that makes the certificate information through an interface like PKCS #11.

The PingDirectory Server supports file-based key stores using the Java KeyStore (JKS) and PKCS #12 formats and also hardware security modules that are accessible through PKCS #11. It does not directly support a key store format that is comprised of individual certificate and private key files, but the manage-certificates tool can be used to import these files into a JKS or PKCS #12 key store.

A key store is a collection of entries, each of which is identified by an alias, sometimes called a nickname. Key stores can have three types of entries:

Private key entries

Contain both a certificate chain and a private key. When a server is performing TLS negotiation, it uses a private key entry to obtain the certificate chain to present to the client, and it can use the private key from that same entry for its key agreement processing. Similarly, if a client presents its own certificate chain to the server, then it uses a private key entry for that.

Trusted certificate entries

Contain a single certificate without a private key. As the name implies, these entries are primarily intended to be used in the course of determining whether to trust a certificate chain that has been presented during TLS negotiation.

Secret key entries

Contain just a secret key without any associated certificate. These types of entries are not used for TLS processing, but rather hold symmetric encryption keys or other types of secrets.

The contents of a key store are protected with a password or passphrase, sometimes called a PIN. In some cases, like with JKS key stores, some of the content can be accessible without a password, and a password might only be required when trying to access private keys or secret keys. In other cases, like with PKCS #12 key stores, you might need a password for any interaction with the key store.

Further, private keys can also be protected with an additional password. This is often the same as the key store password, but it can also be different. This could be useful, for example, if a single key store is shared for multiple purposes, and merely having access to the key store isn't sufficient to have access to all of the data that it contains.

A trust store, or truststore, is another name for a key store that is primarily intended for use in determining whether to trust a certificate chain that has been presented. Trust stores generally contain just trusted certificate entries, but there is no requirement for that to be the case.

Java runtime environments typically include a default trust store, often `jre/lib/security/cacerts or lib/security/cacerts`, that is pre-populated with several widely trusted certification authority certificates. When presented with a certificate signed by one of these authorities, this default trust store could allow the certificate to be trusted without any additional configuration. When presented with a self-signed certificate, or when presented with a certificate signed by an issuer that is not in this default trust store, such as a private corporate CA, a separate trust store is needed.

Understanding TLS

TLS provides support for both trust and encryption.

Here are two types of encryption that make up a TLS session:

Symmetric encryption

Uses the same key for both encryption and decryption. Anyone who has the key can decipher encrypted data and can encrypt new data. The most common symmetric algorithms used in modern TLS cipher suites are AES and ChaCha20. Older cipher suites that are no longer considered secure used additional algorithms like DES, 3DES, and RC4.

Asymmetric encryption (also called public key encryption)

Uses different keys for encryption and decryption. Data encrypted with the public key can only be decrypted with the private key, and data encrypted with the private key can only be decrypted with the public key. The most common asymmetric encryption algorithms used in TLS are RSA and elliptic curve (EC).

Symmetric encryption is generally much faster than asymmetric encryption, but it's only secure if the sender and receiver are the only parties that have the symmetric key. Anyone else who has the key could covertly decrypt the information being transferred, and if they can interject themselves into the middle of the communication, they could transparently alter its content. As such, it's only safe to use if the two parties have previously agreed upon a symmetric key that they use only for communication with each other.

The payload data that is transferred in a TLS session is encrypted with a symmetric cipher. However, this can only happen after a negotiation process that allows the client and server to decide upon which symmetric algorithm to use and what key to use with it.

TLS handshake

There are several steps in the TLS negotiation process, which is also called the handshake.

The process varies somewhat based on the TLS version that is ultimately chosen. The basic components of a TLS 1.2 handshake are as follows:

1. The client sends a client hello message, which tells the server the highest version of the TLS protocol that the client supports and what cipher suites it's willing to use. It also includes a set of extensions with additional information, like the address the client is using to communicate with the server, which signature algorithms and elliptic curves the client supports, and whether it supports secure renegotiation.
2. The server returns a server hello message, which tells the client which TLS protocol version it uses and the cipher suite that it has selected. The server can also provide its own extensions to the client.
3. The server sends a certificate message, in which it provides its certificate chain to the client.
4. The server can optionally send a server key exchange message with additional information that the client might need to help it securely derive the same symmetric encryption key that the server comes up with.
5. The server can optionally send a certificate request message, asking the client to present its own certificate chain to the server.
6. The server sends a server hello done message to tell the client that it has completed its hello sequence.
7. The client can optionally send a certificate message to the server with its own certificate chain. It should only do this if the server sent a certificate request.

   ⓘ **Note:**

   Even if the client got such a request, it can decide not to send a certificate chain and in most cases, it won't. It is up to the server to decide whether it will require a client certificate chain, or whether it is merely optional.

   In LDAP, it's very common for the server to ask the client to present a certificate, but to continue with TLS negotiation even if the client doesn't present one. This makes it possible to support authentication methods like SASL EXTERNAL in which the client can use the certificate chain it presented during TLS negotiation as proof of its identity at the LDAP layer.
8. The client derives a symmetric key to use for the remainder of the encrypted processing, and then sends a client key exchange message to the server that includes the information that the server needs to come up with the same key. This is done in such a way that only the client and the server know what that key is, even if someone else can observe all the communication that passes between the client and the server.
9. If the client presented a certificate chain to the server, then it also sends a certificate verify message to prove that it has the private key for the certificate at the head of the chain.
10. The client sends a change cipher spec message to the server, which tells the server that it is going to start encrypting everything else that it sends to the server with the symmetric key that they have agreed upon.
11. The client sends a finished message to the server, indicating that it has completed its portion of the handshake.
12. The server sends a change cipher spec message to the client, which tells the client that it is going to start encrypting everything else that it sends to the client with the agreed-upon symmetric key.
13. The server sends a finished message to the client, indicating that it has completed its portion of the handshake.

TLS 1.3 uses a dramatically different handshake sequence that might only require a single round trip to exchange all the necessary information between the client and the server as opposed to the two round trips that TLS 1.2 uses. It does this by trying to guess the type of key agreement that the server wants to use and sending the necessary information to the server up front instead of waiting to hear from the server.

ⓘ **Note:**

The elimination of an extra round of communication between the client and the server does mean that the server now finishes its portion of the negotiation before the client, whereas it was previously the other way around.

This means that the server has to assume that the client trusts its certificate chain, while it would have previously known that before completing negotiation. This can make certain types of troubleshooting a little more complicated, because the server can log a successful negotiation, only to later find (though a TLS alert) that the client rejected the certificate.

Key agreement

Key agreement processing is a critical component of TLS negotiation because it allows the client and server to pick the symmetric key that is used to encrypt the remainder of the communication, but without letting anyone else know what that key is.

There are a several different key agreement algorithms, but the most common ones use RSA or variants on Diffie-Hellman.

In RSA key exchange, the client generates some random data, encrypts it using the server's public key, and provides that encrypted data to the server. The server uses its private key to decrypt it and get the random data that the client originally generated. Both the client and the server then derive the encryption key from that random data.

In Diffie-Hellman (DH) key exchange, the client and server publicly agree on a pair of mathematically linked numbers, and then each chooses its own secret value. Through a special computation, they come up with a key that is only known to someone who knows one of the secret values. There are a few different variants of the Diffie-Hellman algorithm that are used in key exchange, but ECDHE and DHE are the recommended versions because they use ephemeral keys without any relation to the server's certificate, and of the two, ECDHE is preferred because it is faster and can use smaller keys without compromising security.

If possible, you should prefer ECHDE over DHE, and either of those over RSA. The Diffie-Hellman algorithms provide a substantial benefit over RSA in the form of forward secrecy. Because RSA key exchange uses the server certificate's public key to encrypt data used to generate the symmetric key, if that certificate's private key is ever compromised, then the encryption can be broken. This includes not only communication on new TLS connections, but also any data that might have been captured in the past. On the other hand, the use of ephemeral keys in ECDHE and DHE ensure that even if the certificate's private key is compromised, any encrypted communication is still indecipherable to anyone but the client and server although anyone with the private key could still pretend to be the legitimate server.

The LDAP StartTLS extended operation

In some cases, it might be possible to establish a non-secure connection and then convert it to a secure one. In LDAP, this can be accomplished with the StartTLS extended operation.

In most cases, when a client wants to use TLS, they establish a connection to a port dedicated to its use, like 636, which is the standard port for LDAPS, or 443, which is the standard port for HTTPS, and the client immediately sends a client hello message over that connection to begin the TLS negotiation process.

There are potential advantages to using the StartTLS extended operation over a dedicated LDAPS connection. It does mean that only one port needs to be opened through a firewall for both secure and insecure communication. It also means that a client can use opportunistic encryption, whereby the client can first query the root DSE to determine whether the server supports StartTLS and then secure the connection if possible. This can be useful in cases like following referrals because LDAP URLs do not officially support "ldaps" as a scheme.

However, we recommend using LDAPS to ensure that the communication is always secure, rather than establishing an initially insecure connection and then securing it with the StartTLS extended operation. If you enable support for unencrypted LDAP communication, which is required for StartTLS, then you run the risk that a client might send sensitive data, such as a bind request containing a password, over that

unencrypted connection. The server can be configured to reject any unencrypted communication, but it can't prevent the client from sending the unencrypted request in the first place.

> ⓘ **Note:**
>
> While it is theoretically possible to use StartTLS to temporarily secure a connection and then drop back to using unencrypted LDAP communication, this is not a recommended practice, and the PingDirectory Server does not support it.

Managing certificates

Because certificates are critical to providing secure communication, it is vital to understand how to manage them in the PingDirectory Server.

The manage-certificates tool

The PingDirectory Server offers a manage-certificates tool that can interact with Java KeyStore (JKS) and PKCS #12 key stores.

It is similar to the `keytool` utility that accompanies most Java distributions, but `manage-certificates` is easier to use, provides better usage information, and offers some additional functionality.

Available subcommands

The **`manage-certificates`** tool uses subcommands to indicate which function you want to invoke.

The subcommands that it offers include the following.

Subcommand	Description
`list-certificates`	Lists the certificates in a key store.
`import-certificate`	Imports a certificate into a trusted certificate entry, or imports a certificate chain and private key into a private key entry.
`export-certificate`	Exports a certificate from a key store.
`export-private-key`	Exports a private key from a key store.
`generate-self-signed-certificate`	Generates a self-signed certificate.
`generate-certificate-signing-request`	Generates a certificate signing request that can be provided to a certification authority.
`sign-certificate-signing-request`	Signs a certificate signing request with a specified issuer certificate.
`check-certificate-usability`	Checks a specified certificate in a key store to verify whether it is suitable for use as a listener certificate.
`trust-server-certificate`	Initiates the TLS negotiation process with a specified server to obtain its certificate chain to update a trust store with information needed to trust that chain.

Subcommand	Description
display-certificate-file	Displays the contents of a file containing one or more PEM-encoded or DER-encoded X.509 certificates.
display-certificate-signing-request-file	Displays the contents of a file containing a PEM-encoded or DER-encoded PKCS #10 certificate signing request (CSR).
change-certificate-alias	Changes the alias for an entry in a key store.
change-keystore-password	Changes the password for a key store.
change-private-key-password	Changes the password used to protect the private key for a specified entry in a key store.

Commonly used arguments
Most of the **manage-certificates** subcommands require access to a Java KeyStore (JKS) or PKCS #12 key store. In such cases, you must specify path to that key store using the **--keystore** argument.

If the key store already exists, then the tool automatically detects whether it is a JKS or PKCS #12 key store. If the operation creates a new key store, then you can explicitly specify the type using the **--keystore-type** argument followed by a value of either “JKS” or “PKCS12”. If you do not specify the key store type, a default value of “JKS” is used.

In some cases, you might also be required to provide the password needed to access the key store. For a JKS key store, you might only need to provide a key store password for operations that involve creating a key store or accessing a private key, but you might need to provide the password for all operations involving a PKCS #12 key store. If you need to provide a key store password, there are three ways that you can do so:

- Using the **--keystore-password** argument followed by the clear-text password for the key store.
- Using the **--keystore-password-file** argument followed by the path to a file containing the password for the key store. The file can contain the password in the clear, or it can be encrypted with a definition from the server’s encryption settings database.
- Using the **--prompt-for-keystore-password** argument. If this argument is provided, the tool interactively prompts for the password.

Private keys can also be protected with a different password than the key store itself. If that is the case, then the private key password can be given in one of the following ways:

- Using the **--private-key-password** argument followed by the clear-text password.
- Using the **--private-key-password-file** argument followed by the path to a file containing the clear-text or encrypted password.
- Using the **--prompt-for-private-key-password** argument, which causes the tool to interactively prompt for the password.

Several operations require you to specify which key store entry you want to target. In such cases, you can provide the **--alias** argument followed by the name of the alias for that entry.

Listing the certificates in a key store
Use the **list-certificates** subcommand to list the certificates in a key store.

You must specify the path to the key store file, and possibly the password needed to access the key store. Other options that are available include:

--alias {alias}

Specifies the alias of the certificate to display. If this is not provided, then all certificates are displayed. This may be provided multiple times to list multiple specific certificates.

--display-pem-certificate

Indicates that a PEM-encoded representation of each certificate should also be included as part of the output.

--verbose

Indicates that the listing should include more detailed information about each of the certificates.

For example, the following command demonstrates a basic listing of a key store containing a single certificate chain.

```
$ bin/manage-certificates list-certificates \
     --keystore config/keystore \
     --keystore-password-file config/keystore.pin

Alias:  server-cert (Certificate 1 of 2 in a chain)
Subject DN:  CN=ds1.example.com,O=Example Corp,C=US
Issuer DN:  CN=Example Certification Authority,O=Example Corp,C=US
Validity Start Time:  Saturday, November 9, 2019 at 11:26:09 AM CST (8
 minutes, 15 seconds ago)
Validity End Time:  Sunday, November 8, 2020 at 11:26:09 AM CST (364 days,
 23 hours, 51 minutes, 44 seconds from now)
Validity State:  The certificate is currently within the validity window.
Signature Algorithm:  SHA-256 with ECDSA
Public Key Algorithm:  EC (secP256r1)
SHA-1 Fingerprint:
 42:f8:85:97:bf:88:bc:74:4b:5b:ce:0c:54:43:9b:44:6b:81:23:a3
SHA-256 Fingerprint:
 4f:be:47:ed:36:68:13:38:ba:e8:c0:c5:6c:85:51:97:8b:40:1b:76:10:c0:be:80:15:62:06:96:c5
Private Key Available:  Yes
The certificate has a valid signature.

Alias:  server-cert (Certificate 2 of 2 in a chain)
Subject DN:  CN=Example Certification Authority,O=Example Corp,C=US
Issuer DN:  CN=Example Certification Authority,O=Example Corp,C=US
Validity Start Time:  Saturday, November 9, 2019 at 11:26:08 AM CST (8
 minutes, 16 seconds ago)
Validity End Time:  Friday, November 4, 2039 at 12:26:08 PM CDT (7299 days,
 23 hours, 51 minutes, 43 seconds from now)
Validity State:  The certificate is currently within the validity window.
Signature Algorithm:  SHA-256 with ECDSA
Public Key Algorithm:  EC (secP256r1)
SHA-1 Fingerprint:
 b8:d0:16:9b:5d:f2:e7:a1:80:79:95:a2:64:b5:aa:ad:80:23:64:16
SHA-256 Fingerprint:
 cf:98:2a:66:35:6e:6d:f9:5d:25:c6:68:68:04:5a:a8:88:43:ca:b5:c8:e5:c9:95:09:e9:fc:ab:b9
The certificate has a valid signature.
```

The following is the verbose version of the previous command.

```
$ bin/manage-certificates list-certificates \
     --keystore config/keystore \
     --keystore-password-file config/keystore.pin \
     --verbose

Alias:  server-cert (Certificate 1 of 2 in a chain)
X.509 Certificate Version:  v3
Subject DN:  CN=ds1.example.com,O=Example Corp,C=US
Issuer DN:  CN=Example Certification Authority,O=Example Corp,C=US
```

```
Serial Number:  7b:2d:91:6a:ff:51:4f:7a:19:16:26:4f:ce:cb:cb:31
Validity Start Time:  Saturday, November 9, 2019 at 11:26:09 AM CST (9
 minutes, 48 seconds ago)
Validity End Time:  Sunday, November 8, 2020 at 11:26:09 AM CST (364 days,
 23 hours, 50 minutes, 11 seconds from now)
Validity State:  The certificate is currently within the validity window.
Signature Algorithm:  SHA-256 with ECDSA
Signature Value:

 30:46:02:21:00:cb:d5:5e:45:b2:8a:33:5e:2d:85:23:39:49:d1:3f:8f:dc:f8:9e:2f:f3:

 44:2f:41:0d:69:95:ec:f0:f5:c0:80:02:21:00:ef:8f:32:35:3c:88:f4:89:ed:f3:a6:76:
     bb:92:6c:eb:c6:17:ac:61:dc:67:26:f0:ec:67:90:51:28:a1:d0:d5
Public Key Algorithm:  EC (secP256r1)
Elliptic Curve Public Key Is Compressed:  false
Elliptic Curve X-Coordinate:
 -24253153720011259408467676608081666342358203254369897642016197975874105796326
Elliptic Curve Y-Coordinate:
 48722714538591494552787288916186748185323678082126843165293664643134352536146
Certificate Extensions:
     Subject Key Identifier Extension:
          OID:  2.5.29.14
          Is Critical:  false
          Key Identifier:
               21:ad:b9:7a:15:e4:08:13:05:e1:c2:64:0c:86:aa:9b:f0:4c:fb:a0
     Authority Key Identifier Extension:
          OID:  2.5.29.35
          Is Critical:  false
          Key Identifier:
               01:4b:69:99:93:5f:76:51:39:95:61:cc:a9:a8:cb:16:f2:0f:8c:c8
     Subject Alternative Name Extension:
          OID:  2.5.29.17
          Is Critical:  false
          DNS Name:  ds1.example.com
          DNS Name:  ds.example.com
          DNS Name:  ldap.example.com
     Key Usage Extension:
          OID:  2.5.29.15
          Is Critical:  false
          Key Usages:
               Digital Signature
               Key Encipherment
               Key Agreement
     Extended Key Usage Extension:
          OID:  2.5.29.37
          Is Critical:  false
          Key Purpose ID:  TLS Server Authentication
          Key Purpose ID:  TLS Client Authentication
SHA-1 Fingerprint:
 42:f8:85:97:bf:88:bc:74:4b:5b:ce:0c:54:43:9b:44:6b:81:23:a3
SHA-256 Fingerprint:
 4f:be:47:ed:36:68:13:38:ba:e8:c0:c5:6c:85:51:97:8b:40:1b:76:10:c0:be:80:15:62:06:96:c5
Private Key Available:  Yes
The certificate has a valid signature.

Alias:  server-cert (Certificate 2 of 2 in a chain)
X.509 Certificate Version:  v3
Subject DN:  CN=Example Certification Authority,O=Example Corp,C=US
Issuer DN:  CN=Example Certification Authority,O=Example Corp,C=US
Serial Number:  43:b7:bb:0c:82:58:42:d8:06:fc:2a:f6:04:e8:2e:8c
Validity Start Time:  Saturday, November 9, 2019 at 11:26:08 AM CST (9
 minutes, 49 seconds ago)
Validity End Time:  Friday, November 4, 2039 at 12:26:08 PM CDT (7299 days,
 23 hours, 50 minutes, 10 seconds from now)
```

```
Validity State:  The certificate is currently within the validity window.
Signature Algorithm:  SHA-256 with ECDSA
Signature Value:

 30:45:02:21:00:b9:87:50:5d:b7:6a:19:82:99:9b:aa:f1:5d:25:a1:90:3c:17:9d:7f:f5:

 7f:8d:06:b4:57:41:9e:15:c6:5a:af:02:20:0c:00:5e:17:bf:ca:bf:0b:ff:db:9f:dc:55:
     ad:35:eb:df:f6:37:4e:23:83:36:88:d2:cc:7d:9e:23:da:78:28
Public Key Algorithm:  EC (secP256r1)
Elliptic Curve Public Key Is Compressed:  false
Elliptic Curve X-Coordinate:
 -20753103001920939059800335367415761738764700353772539765405069978726324039 64
Elliptic Curve Y-Coordinate:
 67079356503908427292378918440889412002659485731683570737365127953554508553 73
Certificate Extensions:
     Subject Key Identifier Extension:
          OID:  2.5.29.14
          Is Critical:  false
          Key Identifier:
               01:4b:69:99:93:5f:76:51:39:95:61:cc:a9:a8:cb:16:f2:0f:8c:c8
     Basic Constraints Extension:
          OID:  2.5.29.19
          Is Critical:  false
          Is CA:  true
          Path Length Constraint:  0
     Key Usage Extension:
          OID:  2.5.29.15
          Is Critical:  false
          Key Usages:
               Key Cert Sign
               CRL Sign
SHA-1 Fingerprint:
 b8:d0:16:9b:5d:f2:e7:a1:80:79:95:a2:64:b5:aa:ad:80:23:64:16
SHA-256 Fingerprint:
 cf:98:2a:66:35:6e:6d:f9:5d:25:c6:68:68:04:5a:a8:88:43:ca:b5:c8:e5:c9:95:09:e9:fc:ab:b9
The certificate has a valid signature.
```

Generating self-signed certificates

A self-signed certificate claims itself as its own issuer, so it is a straightforward process to create one. It's convenient for testing, but because no clients trust it by default, it's not recommended as a listener certificate for production use.

The manage-certificates tool offers a generate-self-signed-certificate subcommand that can be used to create one. In addition to the arguments used to provide information about the key store, certificate alias, and optional private key password, the following arguments are also available.

Argument	Description
`--subject-dn <subject>`	The subject distinguished name (DN) for the certificate to create. This must be provided.
`--days-valid <number>`	The number of days that the certificate is valid. If this is not provided, a default value of 365 is used.

Argument	Description
`--validity-start-time <timestamp>`	A timestamp that indicates when the certificate should begin its validity window. The value should be in the form YYYYMMDDhhmmss (for example, 20190102030405 indicates January 2, 2019 at 3:04:05 a.m.) and is assumed to be in the local time zone. If this is not provided, then the current time is used.
`--key-algorithm <name>`	The name of the algorithm to use to generate the key pair. For a listener certificate, the value is typically either “RSA” or “EC”. This cannot be used in conjunction with the **`--replace-existing-certificate`** argument. If neither that argument nor this argument is provided, a default value of “RSA” is used.
`--key-size-bits <number>`	The length of the key to generate, in bits. This must be provided if the **`--key-algorithm`** argument is also given, and it must not be provided if the **`--replace-existing-certificate`** argument is given. Typical key sizes are 2048 or 4096 bits when using an RSA key, and 256 or 384 bits for an elliptic curve key. If a default RSA key is used and this argument is not provided, then a default key size of 2048 bits is used.
`--signature-algorithm <name>`	The name of the algorithm to use to sign the certificate. Typical signature algorithms are SHA256withRSA for certificates with RSA keys, or SHA256withECDSA for certificates with elliptic curve keys. This must not be provided if the **`--replace-existing-certificate`** argument is used, but it must be given if the **`--key-algorithm`** argument is used to specify an algorithm other than RSA. If a default key algorithm is used and this argument is not provided, then a default value of SHA256withRSA is used.
`--replace-existing-certificate`	Indicates that the new certificate should replace an existing certificate in the key store (in the same alias) and that the key for that certificate should be re-used.
`--inherit-extensions`	Indicates that when replacing an existing certificate, the new certificate should have the same set of extensions as the existing certificate. If the **`--replace-existing-certificate`** argument is provided but this argument is omitted, then the new certificate only has the arguments that are explicitly provided.

Argument	Description
`--subject-alternative-name-dns <name>`	Indicates that the certificate should have a subject alternative name extension with the provided DNS name. The given name should be fully qualified, although it can contain an asterisk as a wildcard in the leftmost component. This argument can be provided multiple times if multiple DNS names are to be included in the subject alternative name extension.
`--subject-alternative-name-ip-address <address>`	Indicates that the certificate should have a subject alternative name extension with the provided IP address. The given address must be a valid IPv4 or IPv6 address, with no wildcards allowed. This argument can be provided multiple times if multiple IP addresses are to be included in the subject alternative name extension.
`--subject-alternative-name-email-address <address>`	Indicates that the certificate should have a subject alternative name extension with the provided email address. This argument can be provided multiple times if multiple email addresses are to be included in the subject alternative name extension.
`--subject-alternative-name-uri <uri>`	Indicates that the certificate should have a subject alternative name extension with the provided URI. This argument can be provided multiple times if multiple URIs are to be included in the subject alternative name extension.
`--subject-alternative-name-oid <oid>`	Indicates that the certificate should have a subject alternative name extension with the provided object identifier (OID). The given value must be a valid OID. This argument can be provided multiple times if multiple OIDs are to be included in the subject alternative name extension.
`--basic-constraints-is-ca <value>`	Indicates that the certificate should have a basic constraints extension with the specified value (which must be either “true” or “false”) for the flag indicating whether the certificate should be considered a certification authority and can therefore be used to sign other certificates. This should be present with a value of “true” for root and intermediate CA certificates. It can optionally be present with a value of “false” for end-entity certificates. It’s probably a good idea for it to be present with a value of “false” for self-signed certificates to indicate that it should not be considered a CA certificate.

Argument	Description
`--basic-constraints-maximum-path-length <number>`	Indicates that the basic constraints extension should include a path length constraint element with the specified value. This can only be used if **`--basic-constraints-is-ca`** is provided with a value of true. A path length constraint value of zero indicates that the certificate can only be used to issue end-entity certificates. If it has a value of one, then it can be used to sign intermediate CA certificates that can only be used to sign end-entity certificates (although it could also be used to sign end-entity certificates). If it has a value that is greater than one, then that means there can be that many intermediate CA certificates between it and the end-entity certificate at the head of the chain.
`--key-usage <value>`	Indicates that the certificate should have a key usage extension with the specified value. Allowed values include “digital-signature”, “non-repudiation”, “key-encipherment”, “data-encipherment”, “key-agreement”, “key-cert-sign”, “crl-sign”, “encipher-only”, and “decipher-only”. This argument can be provided multiple times to include multiple key usages.
`--extended-key-usage <value>`	Indicates that the certificate should have an extended key usage extension with the specified value. Allowed values include “server-auth”, “client-auth”, “code-signing”, “email-protection”, “time-stamping”, and “ocsp-signing”. This argument can be provided multiple times to include multiple extended key usages.

For example, a command like the following can be used to generate a self-signed server certificate.

```
$ bin/manage-certificates generate-self-signed-certificate \
     --keystore config/keystore \
     --keystore-password-file config/keystore.pin \
     --keystore-type JKS \
     --alias server-cert \
     --subject-dn "CN=ds.example.com,O=Example Corp,C=US" \
     --key-algorithm EC \
     --key-length-bits 256 \
     --signature-algorithm SHA256withECDSA \
     --subject-alternative-name-dns ds.example.com \
     --subject-alternative-name-dns ds1.example.com \
     --subject-alternative-name-ip-address 1.2.3.4 \
     --key-usage digital-signature \
     --key-usage key-encipherment \
     --key-usage key-agreement \
     --extended-key-usage server-auth \
     --extended-key-usage client-auth

Successfully created a new JKS keystore.

Successfully generated the following self-signed certificate:
Subject DN:  CN=ds.example.com,O=Example Corp,C=US
```

```
Issuer DN:  CN=ds.example.com,O=Example Corp,C=US
Validity Start Time:  Monday, January 27, 2020 at 03:40:13 PM CST (0 seconds
 ago)
Validity End Time:  Tuesday, January 26, 2021 at 03:40:13 PM CST (364 days,
 23 hours, 59 minutes, 59 seconds from now)
Validity State:  The certificate is currently within the validity window.
Signature Algorithm:  SHA-256 with ECDSA
Public Key Algorithm:  EC (secP256r1)
SHA-1 Fingerprint:
 4f:41:82:7f:08:e9:d8:05:8c:19:8b:3e:5b:bc:59:98:d3:15:71:3a
SHA-256 Fingerprint:
 76:e6:8e:c5:c8:8d:27:ce:2b:85:b9:8c:9d:49:3c:06:f4:40:f1:d0:70:67:39:24:fc:31:bc:f8:51
```

Generating certificate signing requests

A certificate signing request (CSR) contains all of the information needed for a certification authority to issue a certificate.

The request format (also known as PKCS #10) is defined in *RFC 2986* and includes the following elements:

- The certificate signing request version.
- The requested subject distinguished name (DN) for the certificate.
- The public key for the requested certificate.
- The requested set of extensions for the certificate.
- A signature that proves the requester has the private key for the given public key.

The **`manage-certificates generate-certificate-signing-request`** command can be used to create a certificate signing request. It generates a public and private key pair and store it in a key store with a given alias, and it also outputs the certificate signing request to the terminal and optionally write it to a file. Because a certificate signing request contains many of the same elements as a certificate, the command to generate one takes most of the same arguments as for generating a self-signed certificate. However, the following arguments are not available when generating a CSR:

- `--replace-existing-certificate`
- `--days-valid` *<number>*
- `--validity-start-time` *<timestamp>*

The following arguments are available when generating a certificate signing request but not a self-signed certificate.

Argument	Description
`--output-file <path>`	The path to a file to which the certificate signing request should be written. If this is not provided, then the request is only written to the terminal in PEM form.
`--output-format <value>`	The format to use when writing the certificate signing request. The value can be either PEM or DER, but the DER format can only be used in conjunction with the `--output-file` argument. If this argument is not provided, the PEM format is used by default.
`--use-existing-key-pair`	Indicates that the certificate signing request should use a key pair that already exists in the key store with the given alias rather than generating a new key pair, in which case the given alias must not already be in use in the key store.

For example, a command like the following can be used to create a certificate signing request.

```
$ bin/manage-certificates generate-certificate-signing-request \
     --output-file ds1-cert.csr \
     --output-format PEM \
     --keystore config/keystore \
     --keystore-password-file config/keystore.pin \
     --keystore-type JKS \
     --alias server-cert \
     --subject-dn "CN=ds.example.com,O=Example Corp,C=US" \
     --key-algorithm EC \
     --key-length-bits 256 \
     --signature-algorithm SHA256withECDSA \
     --subject-alternative-name-dns ds.example.com \
     --subject-alternative-name-dns ds1.example.com \
     --subject-alternative-name-ip-address 1.2.3.4 \
     --key-usage digital-signature \
     --key-usage key-encipherment \
     --key-usage key-agreement \
     --extended-key-usage server-auth \
     --extended-key-usage client-auth

Successfully created a new JKS keystore.

Successfully generated the key pair to use for the certificate signing
request.

Successfully wrote the certificate signing request to file
'/ds/build/package/PingDirectory/ds1-cert.csr'.
```

The contents of the resulting certificate signing request file can be provided to a certification authority to be signed, and the resulting signed certificate can be imported into the key store as described in *Importing signed and trusted certificates*.

You can also print out the contents of a certificate signing request file using the `display-certificate-signing-request-file` subcommand. This subcommand only supports a couple of arguments.

Argument	Description
`--certificate-signing-request-file` *`<path>`*	The path to the file containing the certificate signing request to be displayed.
`--verbose`	Indicates that the command should display verbose information about the request rather than just a basic set of information.

For example, the following demonstrates the basic output from the command.

```
$ bin/manage-certificates display-certificate-signing-request-file \
     --certificate-signing-request-file ds1-cert.csr

PKCS #10 Certificate Signing Request Version:  v1
Subject DN:  CN=ds.example.com,O=Example Corp,C=US
Signature Algorithm:  SHA-256 with ECDSA
Public Key Algorithm:  EC (secP256r1)
```

The following demonstrates the verbose output.

```
$ bin/manage-certificates display-certificate-signing-request-file \
     --certificate-signing-request-file ds1-cert.csr \
     --verbose
```

```
PKCS #10 Certificate Signing Request Version:  v1
Subject DN:  CN=ds.example.com,O=Example Corp,C=US
Signature Algorithm:  SHA-256 with ECDSA
Signature Value:

 30:45:02:20:46:31:be:9e:6d:2f:0e:e3:d0:80:5c:88:ef:da:86:07:fd:15:b7:62:83:45:

 39:0a:c9:f2:f9:17:eb:08:94:ff:02:21:00:c8:bd:df:57:fa:ea:8c:04:df:c5:27:76:e5:
  b3:3b:4f:df:ec:d3:e4:09:5b:c0:6c:7b:86:39:ec:d0:0e:c1:64
Public Key Algorithm:  EC (secP256r1)
Elliptic Curve Public Key Is Compressed:  false
Elliptic Curve X-Coordinate:
 208628537904757963197889471667098239762296638799662436502070112279302432211 33
Elliptic Curve Y-Coordinate:
 479697739226644990505743464941788269420922508654777168408919906254139602120 95
Certificate Extensions:
  Subject Key Identifier Extension:
       OID:  2.5.29.14
       Is Critical:  false
       Key Identifier:
            f2:de:fd:bf:d3:2f:96:ef:01:70:2d:0e:85:f5:fb:17:d5:a0:9e:67
  Subject Alternative Name Extension:
       OID:  2.5.29.17
       Is Critical:  false
       DNS Name:  ds.example.com
       DNS Name:  ds1.example.com
       IP Address:  1.2.3.4
  Key Usage Extension:
       OID:  2.5.29.15
       Is Critical:  false
       Key Usages:
            Digital Signature
            Key Encipherment
            Key Agreement
  Extended Key Usage Extension:
       OID:  2.5.29.37
       Is Critical:  false
       Key Purpose ID:  TLS Server Authentication
       Key Purpose ID:  TLS Client Authentication
```

Importing signed and trusted certificates

Use the manage-certificates import-certificate command to import certificates into a key store.

There are three primary uses for this command:

- To import a certificate that has been signed by a certification authority into the key store in which the key pair was generated. It is imported into a private key entry, and it should be imported as a certificate chain rather than just the end-entity certificate.
- To import a trusted issuer certificate into a trust store. It is imported into a trusted certificate entry and is a single certificate rather than a chain.
- To import a certificate chain along with the private key for the end-entity certificate. This can be used to import certificates that were generated through some other library like OpenSSL.

In addition to the arguments used to provide information about the key store and the alias into which the certificate (or certificate chain), this command accepts the following arguments.

Argument	Description
`--certificate-file <path>`	The path to the file containing the certificate to be imported. The certificate can be in either PEM or DER format, and it can be a single certificate or a certificate chain. This argument can also be provided multiple times when importing a certificate chain if the certificates in the chain are in separate files.
`--private-key-file <path>`	The path to a file containing the private key that corresponds to the certificate at the head of the chain that is being imported. The private key can be in either PEM or DER format.
`--no-prompt`	Indicates that the certificate should be imported without prompting for confirmation. By default, a summary of the certificate is displayed, and you must confirm that you actually want to import it.

For example, you can use the following command to import a signed certificate into the key store used to generate the certificate signing request.

```
$ bin/manage-certificates import-certificate \
     --keystore config/keystore \
     --keystore-password-file config/keystore.pin \
     --alias server-cert \
     --certificate-file ds1-cert.pem \
     --certificate-file ca-cert.pem

The following certificate chain will be imported into the keystore into
 alias 'server-cert', preserving the existing private key
associated with that alias:

Subject DN:  CN=ds.example.com,O=Example Corp,C=US
Issuer DN:  CN=Example Root CA,O=Example Corp,C=US
Validity Start Time:  Sunday, November 10, 2019 at 09:09:23 PM CST (4
 minutes, 16 seconds ago)
Validity End Time:  Monday, November 9, 2020 at 09:09:23 PM CST (364 days,
 23 hours, 55 minutes, 43 seconds from now)
Validity State:  The certificate is currently within the validity window.
Signature Algorithm:  SHA-256 with ECDSA
Public Key Algorithm:  EC (secP256r1)
SHA-1 Fingerprint:
 02:51:25:43:3e:68:f5:71:36:e3:5d:df:74:de:f6:a1:5a:db:0f:eb
SHA-256 Fingerprint:
 1d:b5:eb:3c:f5:ff:bf:79:a2:a5:86:b8:e4:33:76:4d:d7:50:dc:a4:34:95:37:be:89:45:86:1f:5d

Subject DN:  CN=Example Root CA,O=Example Corp,C=US
Issuer DN:  CN=Example Root CA,O=Example Corp,C=US
Validity Start Time:  Sunday, November 10, 2019 at 09:00:07 PM CST (13
 minutes, 32 seconds ago)
Validity End Time:  Saturday, November 5, 2039 at 10:00:07 PM CDT (7299
 days, 23 hours, 46 minutes, 27 seconds from now)
Validity State:  The certificate is currently within the validity window.
Signature Algorithm:  SHA-256 with ECDSA
Public Key Algorithm:  EC (secP384r1)
SHA-1 Fingerprint:
 0e:5c:21:c9:a5:36:0a:24:eb:aa:55:b6:a5:94:0e:e0:56:03:22:e6
```

```
SHA-256 Fingerprint:
 77:cf:66:d7:3c:8a:fd:67:2d:b7:36:fd:60:1d:ca:eb:1b:03:b1:12:7b:10:1f:26:05:b7:b9:0d:02

Do you want to import this certificate chain into the keystore? yes

Successfully imported the certificate chain.
```

Although the tool displays information about the certificates to be imported if you don't provide the `--no-prompt` argument, you might want to see more information about the certificate before it is imported. You can do so with the `display-certificate-file` subcommand, which offers the following arguments.

Argument	Description
`--certificate-file` *<path>*	The path to the file containing the certificate to view.
`--verbose`	Indicates that verbose information about the certificate should be displayed.

The output of this subcommand has the same format and content as the `list-certificates` subcommand.

Exporting certificates

The `export-certificate` subcommand can be used to export a single certificate or a certificate chain from a key store to a file in either PEM or DER format.

It supports the usual arguments about the key store and the certificate alias, as well as the following additional arguments.

Argument	Description
`--output-file` *<path>*	The path to the file to which the exported certificates will be written. If this is not provided, then the certificates are written to standard output rather than a file.
`--output-format` *<format>*	The format in which the exported certificates are written. The value can be one of PEM or DER, but the DER format can only be used if the output is to be written to a file. If this is not provided, PEM is used as the default format.
`--export-certificate-chain`	Indicates that a certificate chain should be exported rather than just the end-entity certificate.
`--separate-file-per-certificate`	Indicates that a separate output file should be used for each certificate that is exported rather than placing them all in one file. If this argument is provided and multiple certificates are to be exported, then ".1" is appended to the path for the indicated output file for the first certificate in the chain, ".2" is appended for the second certificate, and so on.

For example, something like the following could be used to export a certificate chain.

```
$ bin/manage-certificates export-certificate \
     --keystore config/keystore \
     --keystore-password-file config/keystore.pin \
```

```
     --alias server-cert \
     --output-file server-cert.pem \
     --output-format PEM \
     --export-certificate-chain \
     --separate-file-per-certificate

Successfully exported the following certificate to '/ds/server-cert.pem.1':
Subject DN:  CN=ds.example.com,O=Example Corp,C=US
Issuer DN:  CN=Example Root CA,O=Example Corp,C=US
Validity Start Time:  Sunday, November 10, 2019 at 09:09:23 PM CST (3 hours,
 26 minutes, 23 seconds ago)
Validity End Time:  Monday, November 9, 2020 at 09:09:23 PM CST (364 days,
 20 hours, 33 minutes, 36 seconds from now)
Validity State:  The certificate is currently within the validity window.
Signature Algorithm:  SHA-256 with ECDSA
Public Key Algorithm:  EC (secP256r1)
SHA-1 Fingerprint:
 02:51:25:43:3e:68:f5:71:36:e3:5d:df:74:de:f6:a1:5a:db:0f:eb
SHA-256 Fingerprint:
 1d:b5:eb:3c:f5:ff:bf:79:a2:a5:86:b8:e4:33:76:4d:d7:50:dc:a4:34:95:37:be:89:45:86:1f:5d

Successfully exported the following certificate to '/ds/server-cert.pem.2':
Subject DN:  CN=Example Root CA,O=Example Corp,C=US
Issuer DN:  CN=Example Root CA,O=Example Corp,C=US
Validity Start Time:  Sunday, November 10, 2019 at 09:00:07 PM CST (3 hours,
 35 minutes, 39 seconds ago)
Validity End Time:  Saturday, November 5, 2039 at 10:00:07 PM CDT (7299
 days, 20 hours, 24 minutes, 20 seconds from now)
Validity State:  The certificate is currently within the validity window.
Signature Algorithm:  SHA-256 with ECDSA
Public Key Algorithm:  EC (secP384r1)
SHA-1 Fingerprint:
 0e:5c:21:c9:a5:36:0a:24:eb:aa:55:b6:a5:94:0e:e0:56:03:22:e6
SHA-256 Fingerprint:
 77:cf:66:d7:3c:8a:fd:67:2d:b7:36:fd:60:1d:ca:eb:1b:03:b1:12:7b:10:1f:26:05:b7:b9:0d:02
```

The `export-certificate` subcommand only exports the public portion of a certificate and does not include its private key. If you want to export the private key, then you can use the `export-private-key` subcommand, which supports the following arguments in addition to the usual key store and alias arguments.

Argument	Description
`--output-file` *<path>*	The path to the file to which the exported private key is written. If this is not provided, then the key is written to standard output rather than a file.
`--output-format` *<format>*	The format in which the exported private key is written. The value can be one of PEM or DER, but the DER format can only be used if the output is to be written to a file. If this is not provided, PEM is used as the default format.

The following example includes some of the arguments defined previously.

```
$ bin/manage-certificates export-private-key \
  --keystore config/keystore \
  --keystore-password-file config/keystore.pin \
  --alias server-cert \
  --output-file server-cert-key.pem \
  --output-format PEM
```

```
Successfully exported the private key.
```

Using manage-certificates as a simple certification authority

If your server instances need to support an arbitrary or unknown set of clients, then it's probably best to configure them with certificates from a trusted issuer, like a commercial authority or the free Let's Encrypt service.

However, if you control all of the clients that will access the servers, then you might want to create your own internal certification authority. This allows you to have a common issuer for all servers so that clients just need to trust certificates signed by that issuer. There are commercial and open-source software packages that provide full-featured certification authority functionality, but you can also use the manage-certificates tool to create a certificate authority (CA) certificate and use it to sign certificate signing requests.

The first thing that you need to do is to create a certification authority certificate. This is a self-signed certificate that should have a key usage extension that includes at least the KeyCertSign usage, and a basic constraints extension that indicates that it's a CA certificate. If you don't plan to have an intermediate CA certificate, then the basic constraints extension should have a path length constraint of zero. If you do plan to have an intermediate CA certificate, then the path length constraint should be one to account for it. The CA certificate should also have a long life span because any certificates that it signs is only valid as long as all certificates in the chain are valid.

For example, you can use the following to create a new root CA certificate.

```
$ bin/manage-certificates generate-self-signed-certificate \
     --keystore /ca/root-ca-keystore \
     --keystore-password-file /ca/root-ca-keystore.pin \
     --keystore-type JKS \
     --alias root-ca-cert \
     --subject-dn "CN=Example Root CA,O=Example Corp,C=US" \
     --days-valid 7300 \
     --key-algorithm RSA \
     --key-size-bits 4096 \
     --signature-algorithm SHA256withRSA \
     --basic-constraints-is-ca true \
     --basic-constraints-maximum-path-length 1 \
     --key-usage key-cert-sign \
     --key-usage crl-sign

Successfully created a new JKS keystore.

Successfully generated the following self-signed certificate:
Subject DN:  CN=Example Root CA,O=Example Corp,C=US
Issuer DN:  CN=Example Root CA,O=Example Corp,C=US
Validity Start Time:  Monday, January 27, 2020 at 03:47:29 PM CST (0 seconds
 ago)
Validity End Time:  Sunday, January 22, 2040 at 03:47:29 PM CST (7299 days,
 23 hours, 59 minutes, 59 seconds from now)
Validity State:  The certificate is currently within the validity window.
Signature Algorithm:  SHA-256 with RSA
Public Key Algorithm:  RSA (4096-bit)
SHA-1 Fingerprint:
 bc:8e:5b:30:52:ec:03:63:b4:9a:aa:1a:45:a0:fc:84:49:dd:e8:64
SHA-256 Fingerprint:
 d5:47:06:cd:a2:95:42:61:1f:c7:aa:04:16:1e:c1:70:41:c4:44:48:bf:74:20:5f:1c:61:e2:aa:40
```

You should then export the public portion of that root CA certificate so that you have it for reference, and so that it can be imported as a standalone certificate into trust stores and as part of a certificate chain when importing signed certificates.

If you want to create an intermediate CA certificate, then you should create a new certificate signing request for that certificate. It should essentially use the same settings as for the root CA, although if we assume that there is only a single intermediate CA, then its basic constraints extension should have a path length constraint of zero rather than one to indicate that it can only be used to sign end-entity certificates and cannot itself create subordinate CA certificates. That certificate signing request can be created with a command like the following.

```
$ bin/manage-certificates generate-certificate-signing-request \
     --keystore /ca/intermediate-ca-keystore \
     --keystore-password-file /ca/intermediate-ca-keystore.pin \
     --keystore-type JKS \
     --alias intermediate-ca-cert \
     --subject-dn "CN=Example Intermediate CA,O=Example Corp,C=US" \
     --key-algorithm RSA \
     --key-size-bits 4096 \
     --signature-algorithm SHA256withRSA \
     --basic-constraints-is-ca true \
     --basic-constraints-maximum-path-length 0 \
     --key-usage key-cert-sign \
     --key-usage crl-sign \
     --output-file /ca/intermediate-ca-cert.csr \
     --output-format PEM

Successfully created a new JKS keystore.

Successfully generated the key pair to use for the certificate signing
request.

Successfully wrote the certificate signing request to file
'/ca/intermediate-ca-cert.csr'.
```

Next, we need to use the root CA certificate to sign the certificate signing request for the intermediate CA certificate. That can be done with the sign-certificate-signing-request subcommand, which takes most of the same arguments as generating a self-signed certificate. The primary differences include:

- The key store arguments provided should be for the key store containing the certificate to use to sign the request. The `--signing-certificate-alias` argument must be used to specify the name of the certificate to use to sign the request.
- You must provide a `--request-input-file argument` to specify the path to the file containing the certificate signing request file to generate.
- You can optionally provide a `--certificate-output-file` argument to specify the path to the file to which the signed certificate should be written. If this argument is omitted, then the PEM representation of the certificate is written to standard output.
- You can optionally provide an `--output-format` argument to specify the format (PEM or DER) in which the certificate should be written to the output file.
- You can optionally use the `--subject-dn` argument to specify the subject that should be used for the signed certificate, but you can omit it to indicate that the subject DN from the certificate signing request should be used.
- You cannot specify the key algorithm or key length, because the requester generated the key. You can, however, use the `--signature-algorithm` argument to specify the name of the signature algorithm.
- The `--include-requested-extensions` argument can be used to indicate that the signed certificate should include all of the extensions listed in the certificate signing request. If this argument is not provided, then you must explicitly specify the set of extensions that should be included.

For example, you can use a command like the following to sign the certificate signing request for an intermediate CA certificate.

```
$ bin/manage-certificates sign-certificate-signing-request \
     --keystore /ca/root-ca-keystore \
```

```
     --keystore-password-file /ca/root-ca-keystore.pin \
     --signing-certificate-alias root-ca-cert \
     --days-valid 7300 \
     --include-requested-extensions \
     --request-input-file /ca/intermediate-ca-cert.csr \
     --certificate-output-file /ca/intermediate-ca-cert.pem \
     --output-format PEM

Read the following certificate signing request:

PKCS #10 Certificate Signing Request Version:  v1
Subject DN:  CN=Example Intermediate CA,O=Example Corp,C=US
Signature Algorithm:  SHA-256 with RSA
Public Key Algorithm:  RSA (4096-bit)

Do you really want to sign this request? yes

Successfully wrote the signed certificate to file
'/ca/intermediate-ca-cert.pem'.
```

Once you have the intermediate CA certificate, you should create secure, offline backups of the root CA certificate, and then remove it (or at least its private key) from all systems. All of the end-entity certificates should be signed with the intermediate CA certificate, and that process is identical to the example given above. The only reason that you should need to pull the root CA certificate out of cold storage is if you need to sign another intermediate CA certificate.

The PingDirectory Server's use of certificates

The PingDirectory Server uses two main types of certificates: listener certificates and an inter-server certificate.

Listener certificates

Listener certificates are those that are used to secure communication through TLS.

Whenever a client initiates TLS negotiation with the server, it presents a certificate chain to the client, and the certificate at the head of that chain will be a listener certificate. The client must decide whether to trust the certificate chain, so it is beneficial to ensure that it is signed by an issuer that the client is likely to trust or at least, that the client can be easily configured to trust.

While you can create self-signed certificates with very long life spans, any certificate that you have signed by a certification authority is likely to have a relatively short life span. Commercial authorities are only likely to issue certificates that are valid for up to one or two years, and some use much shorter validity windows. For example, the nonprofit Let's Encrypt service, which can be used to obtain trusted certificates for free as long as you can prove that you control the domains for which they are to be issued, only issues certificates that are valid for up to three months.

While short certificate life spans can be inconvenient for administrators who need to replace them, they do offer some security benefits. In particular, because most clients don't make any attempt to check whether a certificate has been revoked, a relatively short validity window minimizes the time that a compromised certificate can be used. And if the process for replacing certificates is automated or least streamlined, then the inconvenience can be kept to a minimum.

Listener certificates are stored in key stores, and those key stores are referenced by key manager providers, which provide the logic and configuration needed to access the key stores. If a server component, for example, a connection handler, needs to have access to a certificate that it presents to a peer during the TLS negotiation process, then that component should reference the key manager provider that points to the key store containing the desired certificate. If the key store has more than one certificate, and if the component referencing it includes a property that specifies the nickname of the certificate to use, then the certificate with that alias is selected. Otherwise, the server leaves it up to the JVM to select a certificate, and which one it chooses is not well defined.

The server also provides trust manager providers that can determine whether to trust any presented certificate chains. A trust manager provider can reference a specified trust store file, but other options include the JVM-default trust store which uses the Java installation's default set of trusted issuers and the blind trust manager provider which blindly trusts any certificate chain that is presented to it. Do not use the blind trust manager in a production environment because it leaves the server vulnerable to impersonation and interception attacks, but it can be convenient in test environments to troubleshoot certain types of problems.

The inter-server certificate
Use the inter-server certificate to authenticate to other server instances in the topology.

Each instance has a unique inter-server certificate that is generated during the setup process. This certificate is not exposed to clients, so there is no need for it to be signed by a trusted issuer. The topology registry, a mirrored portion of the configuration with information about all of the PingDirectory Server instances in the environment, has all of the information that each instance needs to trust the inter-server certificates for all of the other instances.

Inter-server certificates can also be used to protect certain secrets that are shared among servers within the topology, like those used to digitally sign log files, backups, or LDIF exports. It also includes the encryption keys used by reversible password storage schemes.

The inter-server certificate is generated with a very long life span and should not ever need to be replaced unless you suspect that its private key has been compromised.

Replacing listener certificates

Because you might want your listener certificates issued by a certification authority rather than using self-signed certificates, and certification authorities typically use restricted life spans for the certificates that they sign, you might need to replace those certificates on a fairly regular basis.

This includes obtaining a new certificate chain, making any necessary updates to the key manager provider and connection handler configuration, and updating the server instance listener configuration with the new certificate. This is something that can be done manually, but the replace-certificate tool can help automate that process. Updating the server instance listener configuration can be onerous because it should provide information about multiple listener certificates at least during the transitional phase when the new certificate is being installed.

The `replace-certificate` tool offers both interactive and non-interactive modes of operation. The interactive mode is convenient because it walks you through the process of obtaining a new certificate and installing it in the server, while the non-interactive mode is better suited for scripting the process of replacing the certificate. The interactive mode is also useful for learning the tool because it displays the non-interactive commands needed to achieve the same result.

The replace-listener-certificate subcommand handles all the work involved in replacing a listener certificate. In addition to arguments needed to authenticate to the server (for example, `--bindDN` and `--bindPasswordFile`), this subcommand takes arguments that provide information about the key store containing the new certificate, which key and trust manager provider updates should be made, and whether to signal the HTTP connection handler to reload its certificates after the update is complete. The available arguments include.

Argument	Description
`--source-key-store-file` *<path>*	The path to the Java KeyStore (JKS) or PKCS #12 file that contains the private key entry with the new certificate chain. This must be provided.
`--source-key-store-password` *<password>*	The clear-text password needed to access the contents of the source key store.

Argument	Description
`--source-key-store-password-file` *<path>*	The path to a file containing the password needed to access the contents of the source key store. The file can contain the password in the clear, or it can be encrypted with a definition from the server's encryption settings database.
`--source-certificate-alias` *<alias>*	The alias of the private key entry in the source key store that contains the certificate chain for the new listener certificate. If the source key store has more than one private key entry, then this argument must be provided to indicate which one to use.
`--source-private-key-password` *<password>*	The password needed to access the appropriate private key in the source key store. If neither the `--source-private-key-password` nor the `--source-private-key-password-file` argument is provided, then the key store password is also used as the private key password.
`--source-private-key-password-file` *<path>*	The path to a file containing the password needed to access the appropriate private key in the source key store. The file can contain the password in the clear, or it can be encrypted with a definition from the server's encryption settings database. If neither the `--source-private-key-password` nor the `--source-private-key-password-file` argument is provided, then the key store password is also used as the private key password.
`--key-manager-provider` *<name>*	The name of the key manager provider that will be updated to use the new certificate chain. It must be a file-based key manager provider, and it must be enabled. If this argument is not provided, a default value of "JKS" is assumed.
`--trust-manager-provider` *<name>*	The name of the trust manager provider that will be updated with the information needed to trust the new certificate chain. It must be a file-based trust manager provider, and it must be enabled. If neither this argument nor the `--use-jvm-default-trust-manager-provider` argument is provided, then the tool assumes that the trust manager provider has the same name as the key manager provider.
`--use-jvm-default-trust-manager-provider`	Indicates that the server should be configured to use the JVM-default trust manager provider, which trusts certificates signed by issuers in the cacerts trust store provided with the JVM, rather than updating an existing trust manager provider.

Argument	Description
`--target-certificate-alias <alias>`	The alias to use for the new certificate in the key manager provider's key store and also for any appropriate updates in the trust manager provider's trust store. If this is not provided, a default alias of "server-cert" is used. **Note:** If the key manager provider's key store or the trust manager provider's trust store already contains an entry with the given alias, the existing entry will be renamed.
`--reload-http-connection-handler-certificates`	Indicates that the tool should request that the server cause any HTTPS-based connection handlers to reload their certificates so that they will start using the updated certificate. LDAP connection handlers reacts to the change right away and start presenting the new certificate chain during any subsequent TLS negotiations, but HTTPS connection handlers will continue using the former certificate until the connection handler is restarted or until it is specifically asked to reload its certificates. **Note:** Using this option can prevent clients with existing TLS sessions negotiated with the former certificate from being resumed.

The following example demonstrates using the tool in interactive mode to replace the listener certificate chain. The output also includes the non-interactive commands needed to perform the corresponding operations.

```
$ bin/replace-certificate
This tool can be used to replace the listener certificate or the
inter-server certificate for this Directory Server server instance

Which action would you like to perform?

1 - Replace a listener certificate that the server uses forTLS
 communication
2 - Replace the inter-server certificate that the server uses to
 authenticate to other instances in the topology
3 - Purge any retired listener certificates for this server from the
 topology registry
4 - Purge any retired inter-server certificates for this server from the
 topology registry
q - Quit without doing anything

Enter your choice: 1



Enter the DN of the account to use to authenticate to the server
```

```
[cn=Directory Manager]: cn=Directory Manager
Enter the password for that user: {password}

NOTE:  'JKS' is the only key manager provider that is suitable to be
updated with a new listener certificate.  Automatically selecting that
provider


Which trust manager provider do you wish to update with information needed
to trust the new listener certificate?

1 - JKS
d - Use the JVM-default trust manager provider to trust any certificate
 signed by an authority in the JVM's default set of trusted issuers

Enter your choice [1]: 1



How would you like to obtain the new listener certificate?

1 - Generate a new self-signed certificate
2 - Generate a request for a certificate to be signed by a certification
    authority
3 - Use a certificate in an existing key store
q - Quit without doing anything

Enter your choice: 2

Enter the subject DN that you would like to use for the new certificate.
The subject DN typically includes some or all of the following components:

* CN -- The common name for the certificate.  This is typically the
  fully-qualified name (not an IP address) that most clients will use to
  connect to the server (alternate names and IP addresses may be provided
  later).  We strongly recommend including a CN attribute in the
  certificate subject
* OU -- Typically the name of the department or organizational unit that
  manages the server
* O -- Typically the name of the company or organization that manages the
  server
* L -- Typically the name of the city or locality in which the server is
  located
* ST -- Typically the full name (NOT an abbreviation) of the state or
  province in which the server is located
* ST -- Typically the two-character ISO 3166 country code for the country
  in which the server is located

For example, a subject DN might look like 'CN=ds.example.com,OU=Directory
Services,O=Example Corp,L=Austin,ST=Texas,C=US'

Enter the desired subject DN: CN=ds1.example.com,O=Example Corp,C=US

Enter the complete set of resolvable names (not IP addresses) that clients
are expected to use to access the server.  These names will be included in
the certificate's subject alternative name extension

Specific host names are generally preferable, but you may use an asterisk
as a wildcard in the leftmost component that will match any host name in
that component.  For example, '*.example.com' indicates that the
certificate may be used in any server whose fully-qualified name consists of
exactly three components, and in which the last two components are
'example.com'
```

```
The current set of DNS names to include in the set of subject alternative
names is:

* ds1.example.com
* ip6-localhost
* localhost

What would you like to do?

1 - Use the current set of DNS names
2 - Add another DNS name
3 - Remove a specific DNS name
4 - Clear the current set of DNS names
5 - Do not include any subject alternative DNS names in the certificate

Enter your choice [1]: 2

Enter the new DNS name to include: ds.example.com


The current set of DNS names to include in the set of subject alternative
names is:

* ds.example.com
* ds1.example.com
* ip6-localhost
* localhost

What would you like to do?

1 - Use the current set of DNS names
2 - Add another DNS name
3 - Remove a specific DNS name
4 - Clear the current set of DNS names
5 - Do not include any subject alternative DNS names in the certificate

Enter your choice [1]: 1

Enter the complete set of IPv4 and IPv6 addresses that clients are
expected to use to access the server.  These addresses will be included in
the certificate's subject alternative name extension.  Wildcards are not
allowed


The current set of IP addresses to include in the set of subject
alternative names is:

* 0:0:0:0:0:0:0:1
* 10.5.1.133
* 10.5.3.99
* 127.0.0.1
* 127.0.1.1
* 172.30.12.185
* fe80:0:0:0:3957:af69:bd92:6c73
* fe80:0:0:0:ace8:231f:e348:db8d
* fe80:0:0:0:fc94:6eff:fe1d:811d

What would you like to do?

1 - Use the current set of IP addresses
2 - Add another IP address
3 - Remove a specific IP address
4 - Clear the current set of IP addresses
5 - Do not include any subject alternative IP addresses in the certificate
```

```
Enter your choice [1]: 1

Generating a certificate signing request with the following command:

     manage-certificates \
          generate-certificate-signing-request \
          --output-file /ds/tmp/replace-certificate-certificate-signing-
request-6730986100632343057.pem \
          --output-format PEM \
          --keystore /ds/tmp/replace-certificate-temporary-key-
store-281484917294163222.jks \
          --keystore-password-file '*****REDACTED*****' \
          --keystore-type JKS \
          --alias generated-certificate \
          --subject-dn "CN=ds1.example.com,O=Example Corp,C=US" \
          --key-algorithm RSA \
          --key-size-bits 2048 \
          --signature-algorithm SHA256withRSA \
          --key-usage digitalSignature \
          --key-usage keyEncipherment \
          --extended-key-usage server-auth \
          --extended-key-usage client-auth \
          --subject-alternative-name-dns ds.example.com \
          --subject-alternative-name-dns ds1.example.com \
          --subject-alternative-name-dns ip6-localhost \
          --subject-alternative-name-dns localhost \
          --subject-alternative-name-ip-address 0:0:0:0:0:0:0:1 \
          --subject-alternative-name-ip-address 10.5.1.133 \
          --subject-alternative-name-ip-address 10.5.3.99 \
          --subject-alternative-name-ip-address 127.0.0.1 \
          --subject-alternative-name-ip-address 127.0.1.1 \
          --subject-alternative-name-ip-address 172.30.12.185 \
          --subject-alternative-name-ip-address
 fe80:0:0:0:3957:af69:bd92:6c73 \
          --subject-alternative-name-ip-address
 fe80:0:0:0:ace8:231f:e348:db8d \
          --subject-alternative-name-ip-address
 fe80:0:0:0:fc94:6eff:fe1d:811d

Successfully generated the following certificate signing request, which
has also been written to file
'/ds/tmp/replace-certificate-certificate-signing-
request-6730986100632343057.pem':

-----BEGIN CERTIFICATE REQUEST-----
MIIDoTCCAosCAQAwPjELMAkGA1UEBgwCVVMxFTATBgNVBAoMDEV4YW1wbGUgQ29y
cDEYMBYGA1UEAwwPZHMxLmV4YW1wbGUuY29tMIIBIjANBgkqhkiG9w0BAQEFAAOC
AQ8AMIIBCgKCAQEAgYFiIt72o1uHAbqyE3dnMUDvf/tBl6ItzzODkk8oUtF4bRFu
K+RZvz+NzkADVbwIkXOLIIbmRcQZdyPMCZ/gmhzLhwetyu3IeTbPZk682iEf8B5r
8Q/4FwrmoTRuRSrYJ4V9N6lPyrouK2i8G72GA8QiUCG6Cjil4aIhjkPjqbSD9lZF
Nv7BWmsizXaQ/j8I2yRly29JKxp84mO0h6N9npqCIQN/XqdFbFfNERO0h7oBOFpH
J+yepsmbOdQE2Ywbru4TqqizVgsokQSr7oGWSSsS5KwwSPUwjhmsNzIU20UBEjrV
Xa3XiXxBK6aKYWqXlN4N3rxaYZQWbxMJK5wWwQIDAQABoIIBHjCCARoGCSqGSIb3
DQEJDjGCAQswggEHMB0GA1UdDgQWBBT5FA1stRYO99mFHb3BzjMkyRLbBjCBuAYD
VR0RBIGwMIGtgg5kcy5leGFtcGxlLmNvbYIPZHMxLmV4YW1wbGUuY29tgg1pcDYt
bG9jYWxob3N0gglsb2NhbGhvc3SCCm5hdy1zZXJ2YWyHEAAAAAAAAAAAAAAAAAAA
AAGHBAoFAYWHBAoFA2OHBH8AAAGHBH8AAQGHBKweDLmHEP6AAAAAAAAAOVevab2S
bHOHEP6AAAAAAAArOgjH+NI242HEP6AAAAAAAAA/JRu//4dgR0wDAYDVR0PBAUD
AwegADAdBgNVHSUEFjAUBggrBgEFBQcDAQYIKwYBBQUHAwIwCwYJKoZIhvcNAQEL
A4IBAQA0B/QipNqyj0nqJ8lz4xXbE3axtswdosJ140nIzDXNE5TRBpQZg8PgfhI/
argRATg7XuWe4xapgYpVpNBFZOLAKsknc1AjVvjakavprQkRYnhKqH6Delao2Lo+
1KsznomWn4oEoEWZ6+AHtuf50t2LeBc82tQTdijxhqlfIPhVd+spDxMhcdWehPOx
ZZoTyAtiCDfjUhMxh7ez+jAKHwLkiZXwgiPJvRbYdbz3/1Bs2XqIj7mfmYrU6zAr
```

```
Zr+Jo/utfw6ayDwp32+9JtIAROF9GqVGxrJeAam789rVnr+Bh3jK2VdccTNdvi4H
1nPfqr6v6lpMdrcW/chboRETOtl9
-----END CERTIFICATE REQUEST-----


How would you like to import the signed certificate into the key store?

1 - I expect to get the signed certificate back right away.  Walk me
 through the process of importing it into the key store
2 - I will manually import the signed certificate

Enter your choice [1]: 1
Enter the path to the file containing the signed certificate (and
optionally any necessary issuer certificates): /ca/server-cert.pem


NOTICE:  Certificate file '/ca/server-cert.pem' ends with certificate
'CN=ds1.example.com,O=Example Corp,C=US' that was signed by issuer
'CN=Example Intermediate CA,O=Example Corp,C=US', but that issuer
certificate could not be found in the JVM-default trust store

Enter the path to a file containing the
'CN=Example Intermediate CA,O=Example Corp,C=US' certificate (and
optionally its issuer chain): /ca/intermediate-ca-cert.pem


NOTICE:  Certificate file '/ca/intermediate-ca-cert.pem' ends with
certificate 'CN=Example Intermediate CA,O=Example Corp,C=US' that was
signed by issuer 'CN=Example Root CA,O=Example Corp,C=US', but that issuer
certificate could not be found in the JVM-default trust store

Enter the path to a file containing the 'CN=Example Root CA,O=Example
Corp,C=US' certificate (and optionally its issuer chain): /ca/root-ca-
cert.pem



Would you like to request that the server reload any certificates
associated with HTTP connection handlers configured with support for HTTPS
so that they will start using the new certificate right away?  Note that
this may prevent clients from resuming TLS sessions created before the
reload

1 - Yes.  Reload the certificates associated with any HTTPS-enabled HTTP
 connection handlers
2 - No.  Do not reload the certificates.  HTTPS connection handlers will
 continue to use their current certificates until the server (or at
 least the connection handler) is restarted, or until the
 reload-http-connection-handler-certificates tool is run
q - Quit without doing anything else

Enter your choice [1]: 1


About to invoke the following command:

     replace-certificate \
          replace-listener-certificate \
          --bindDN "cn=Directory Manager" \
          --bindPassword '*****REDACTED*****' \
          --key-manager-provider JKS \
          --trust-manager-provider JKS \
          --source-key-store-file /ds/tmp/replace-certificate-temporary-key-
store-281484917294163222.jks \
```

```
          --source-key-store-password-file /ds/config/ads-truststore.pin \
          --source-certificate-alias generated-certificate \
          --reload-http-connection-handler-certificates

Do you want to invoke this command?

1 - Yes, run this replace-certificate command
2 - No.  Quit without doing anything else

Enter your choice [1]: 1
Successfully replaced the listener certificate
```

As part of its processing, the `replace-certificate` tool updates the server instance listener configuration object to include the new listener certificate. It will merge that new certificate with any existing certificates in that configuration object rather than replacing them. If you want to remove any older certificates from the server instance listener configuration, you can do that with the `purge-retired-listener-certificates` subcommand, which doesn't take any arguments other than those needed to authenticate to the server, as in the following example.

```
$ bin/replace-certificate
This tool can be used to replace the listener certificate or the
inter-server certificate for this Directory Server server instance

Which action would you like to perform?

1 - Replace a listener certificate that the server uses for TLS
 communication
2 - Replace the inter-server certificate that the server uses to
 authenticate to other instances in the topology
3 - Purge any retired listener certificates for this server from the
 topology registry
4 - Purge any retired inter-server certificates for this server from the
 topology registry
q - Quit without doing anything

Enter your choice: 3



Enter the DN of the account to use to authenticate to the server
[cn=Directory Manager]: cn=Directory Manager
Enter the password for that user: {password}


About to invoke the following command:

  replace-certificate \
       purge-retired-listener-certificates \
       --bindDN "cn=Directory Manager" \
       --bindPassword '*****REDACTED*****'

Do you want to invoke this command?

1 - Yes, run this replace-certificate command
2 - No.  Quit without doing anything else

Enter your choice [1]: 1

NOTE:  Purging one non-active listener certificate from entry
'cn=ldap-listener-mirrored-config,cn=Server Instance
 Listeners,cn=ds1,cn=Server
Instances,cn=Topology,cn=config'
```

```
Successfully updated entry 'cn=ldap-listener-mirrored-config,cn=Server
 Instance
Listeners,cn=ds1,cn=Server Instances,cn=Topology,cn=config'

Successfully purged one retired listener certificate
```

Replacing the inter-server certificate

The inter-server certificate is only intended for use between instances within the same topology.

It is not exposed to normal clients, so it doesn't need to be trusted. It is generated at install time with along life span, so it should not need to be replaced under normal circumstances. In fact, we discourage replacing the inter-server certificate unless you have reason to suspect that its private key has been compromised.

If you do need to replace the inter-server certificate, the `replace-certificate replace-inter-server-certificate` command can be used to accomplish this. As when replacing a listener certificate, it takes the new inter-server certificate from a provided Java KeyStore (JKS) or PKCS #12 key store, and it makes the necessary updates to the appropriate server key store, and in the case of the inter-server certificate, that is the one in the `config/ads-truststore` file. It also updates the server instance configuration object to include the new inter-server certificate.

> ⓘ **Note:**
>
> The server currently requires the inter-server certificate to use an RSA key pair, and the key size must be at least 2048 bits. Certificates with an elliptic curve key pair are not allowed, nor are certificates with an RSA key smaller than 2048 bits.

Use a self-signed certificate with a long life span so that it does not need to be replaced on a regular basis. Each instance should have its own unique inter-server certificate.

The `replace-inter-server-certificate` subcommand takes a subset of the arguments used in conjunction with the replace-listener-certificate subcommand. The available arguments include:

- `--source-key-store-file <path>--source-key-store-password <password>`
- `--source-key-store-password-file <path>`
- `--source-certificate-alias <alias>`
- `--source-private-key-password <password>`
- `--source-private-key-password-file <path>`

The following example demonstrates the process for replacing the inter-server certificate in interactive mode, but includes the non-interactive command needed to achieve the same result.

```
$ bin/replace-certificate
This tool can be used to replace the listener certificate or the
inter-server certificate for this Directory Server server instance

Which action would you like to perform?

1 - Replace a listener certificate that the server uses for TLS
 communication
2 - Replace the inter-server certificate that the server uses to
 authenticate to other instances in the topology
3 - Purge any retired listener certificates for this server from the
 topology registry
4 - Purge any retired inter-server certificates for this server from the
 topology registry
q - Quit without doing anything

Enter your choice: 2
```

```
WARNING:  The inter-server certificate is used only for the purpose of
authenticating this server instance to other servers in the topology, and
to encrypt some legacy secrets.  It is NOT used to encrypt communication
(listener certificates are used for that purpose), and the trust mechanism
that we use for authenticating inter-server certificates is stronger when
using self-signed certificates than when using certificates signed by a
publicly trusted authority

We strongly discourage replacing the inter-server certificate unless you
believe that the key has been compromised.  Any errors in the process of
replacing the certificate could render the server unable to authenticate
to other instances in the topology, and may interfere with replication or
other forms of inter-server communication, and also the server's ability
to perform certain types of encryption and digital signing

Are you sure you want to replace the inter-server certificate?

1 - No.  Do not replace the inter-server certificate
2 - Yes.  Proceed with replacing the inter-server certificate

Enter your choice [1]: 2



Enter the DN of the account to use to authenticate to the server
[cn=Directory Manager]: cn=Directory Manager
Enter the password for that user: {password}



How would you like to obtain the new inter-server certificate?

1 - Generate a new self-signed certificate
2 - Generate a request for a certificate to be signed by a certification
 authority
3 - Use a certificate in an existing key store.  Note that each server
 instance must have a unique inter-server certificate, and we do not
 recommend using the same certificate as both a listener certificate
 and an inter-server certificate
q - Quit without doing anything

Enter your choice: 1

Enter the subject DN that you would like to use for the new certificate.
The subject DN typically includes some or all of the following components:

* CN -- The common name for the certificate.  This is typically the
  fully-qualified name (not an IP address) that most clients will use to
  connect to the server (alternate names and IP addresses may be provided
  later).  We strongly recommend including a CN attribute in the
  certificate subject
* OU -- Typically the name of the department or organizational unit that
  manages the server
* O -- Typically the name of the company or organization that manages the
  server
* L -- Typically the name of the city or locality in which the server is
  located
* ST -- Typically the full name (NOT an abbreviation) of the state or
  province in which the server is located
* ST -- Typically the two-character ISO 3166 country code for the country
  in which the server is located
```

```
For example, a subject DN might look like 'CN=ds.example.com,OU=Directory
Services,O=Example Corp,L=Austin,ST=Texas,C=US'

Enter the desired subject DN: CN=ds1,O=Example Corp,C=US

Enter the complete set of resolvable names (not IP addresses) that clients
are expected to use to access the server.  These names will be included in
the certificate's subject alternative name extension

Specific host names are generally preferable, but you may use an asterisk
as a wildcard in the leftmost component that will match any host name in
that component.  For example, '*.example.com' indicates that the
certificate may be used in any server whose fully-qualified name consists
of exactly three components, and in which the last two components are
'example.com'

The current set of DNS names to include in the set of subject alternative
names is:

* ds1.example.com
* ip6-localhost
* localhost

What would you like to do?

1 - Use the current set of DNS names
2 - Add another DNS name
3 - Remove a specific DNS name
4 - Clear the current set of DNS names
5 - Do not include any subject alternative DNS names in the certificate

Enter your choice [1]: 1

Enter the complete set of IPv4 and IPv6 addresses that clients are
expected to use to access the server.  These addresses will be included in
the certificate's subject alternative name extension.  Wildcards are not
allowed


The current set of IP addresses to include in the set of subject
alternative names is:

* 0:0:0:0:0:0:0:1
* 10.5.1.133
* 10.5.3.99
* 127.0.0.1
* 127.0.1.1
* 172.30.12.185
* fe80:0:0:0:3957:af69:bd92:6c73
* fe80:0:0:0:ace8:231f:e348:db8d
* fe80:0:0:0:fc94:6eff:fe1d:811d

What would you like to do?

1 - Use the current set of IP addresses
2 - Add another IP address
3 - Remove a specific IP address
4 - Clear the current set of IP addresses
5 - Do not include any subject alternative IP addresses in the certificate

Enter your choice [1]: 1

Generating a self-signed certificate with the following command:
```

```
     manage-certificates \
          generate-self-signed-certificate \
          --keystore /ds/tmp/replace-certificate-temporary-key-
store-12068302381295037387.jks \
          --keystore-password-file '*****REDACTED*****' \
          --keystore-type JKS \
          --alias generated-certificate \
          --subject-dn "CN=ds1,O=Example Corp,C=US" \
          --validity-start-time 20191111120632 \
          --days-valid 7300 \
          --key-algorithm RSA \
          --key-size-bits 2048 \
          --signature-algorithm SHA256withRSA \
          --key-usage digitalSignature \
          --key-usage keyEncipherment \
          --extended-key-usage server-auth \
          --extended-key-usage client-auth \
          --subject-alternative-name-dns ds1.example.com \
          --subject-alternative-name-dns ip6-localhost \
          --subject-alternative-name-dns localhost \
          --subject-alternative-name-ip-address 0:0:0:0:0:0:0:1 \
          --subject-alternative-name-ip-address 10.5.1.133 \
          --subject-alternative-name-ip-address 10.5.3.99 \
          --subject-alternative-name-ip-address 127.0.0.1 \
          --subject-alternative-name-ip-address 127.0.1.1 \
          --subject-alternative-name-ip-address 172.30.12.185 \
          --subject-alternative-name-ip-address
 fe80:0:0:0:3957:af69:bd92:6c73 \
          --subject-alternative-name-ip-address
 fe80:0:0:0:ace8:231f:e348:db8d \
          --subject-alternative-name-ip-address
 fe80:0:0:0:fc94:6eff:fe1d:811d

Successfully generated the self-signed certificate


About to invoke the following command:

     replace-certificate \
          replace-inter-server-certificate \
          --bindDN "cn=Directory Manager" \
          --bindPassword '*****REDACTED*****' \
          --source-key-store-file /ds/tmp/replace-certificate-temporary-key-
store-12068302381295037387.jks \
          --source-key-store-password-file /ds/config/ads-truststore.pin \
          --source-certificate-alias generated-certificate

Do you want to invoke this command?

1 - Yes, run this replace-certificate command
2 - No.  Quit without doing anything else

Enter your choice [1]: 1
Successfully replaced the inter-server certificate
```

The new inter-server certificate is merged with any existing values in the server instance configuration entry. You can use the `purge-retired-inter-server-certificates` subcommand to remove any older values once you are confident that they are no longer needed, as in the following example.

```
$ bin/replace-certificate
This tool can be used to replace the listener certificate or the
inter-server certificate for this Directory Server server instance
```

```
Which action would you like to perform?

1 - Replace a listener certificate that the server uses for TLS
 communication
2 - Replace the inter-server certificate that the server uses to
 authenticate to other instances in the topology
3 - Purge any retired listener certificates for this server from the
 topology registry
4 - Purge any retired inter-server certificates for this server from the
 topology registry
q - Quit without doing anything

Enter your choice: 4



Enter the DN of the account to use to authenticate to the server
[cn=Directory Manager]: cn=Directory Manager
Enter the password for that user: {password}


About to invoke the following command:

  replace-certificate \
       purge-retired-inter-server-certificates \
       --bindDN "cn=Directory Manager" \
       --bindPassword '*****REDACTED*****'

Do you want to invoke this command?

1 - Yes, run this replace-certificate command
2 - No.  Quit without doing anything else

Enter your choice [1]: 1
# Initializing the server's encryption framework...
Successfully purged one retired inter-server certificate from the topology
registry
```

Enabling TLS in the PingDirectory Server

You can enable support for TLS when initially setting up the server. If you already have a server instance without TLS support, you can enable it after the fact.

Enabling TLS support during setup

The easiest way to enable TLS support in the server is to do so during setup. You can do so by either providing a key store containing the certificate you want to use or by having the installer generate a self-signed certificate for you.

If you are running `setup` in interactive mode, then it prompts you for all of the questions needed to configure secure communication.

```
Do you want to enable the Directory Server services (Available State,
Available or Degraded State, Configuration, Consent, Directory REST API,
Documentation, Instance Root File, SCIM2, and Swagger UI) and Administrative
Console over HTTPS?  After setup, you can selectively enable or disable
individual services and applications by configuring the HTTPS Connection
Handler (yes / no) [yes]: yes

On which port should the Directory Server accept connections from HTTPS
clients? [443]: 443
```

```
Do you want to accept unencrypted LDAP connections?

 1)  Do not accept unencrypted LDAP connections
 2)  Accept unencrypted LDAP connections, but require StartTLS to secure all
     communication on those connections
 3)  Accept unencrypted LDAP connections, but optionally allow StartTLS to
     secure communication on those connections
 4)  Accept unencrypted LDAP connections and do not enable support for
     StartTLS

Enter option [3]: 3

On which port should the Directory Server accept connections from LDAP
clients? [389]: 389

Do you want to enable LDAPS? (yes / no) [yes]: yes
On which port should the Directory Server accept connections from LDAPS
clients? [636]: 636

Certificate server options:

 1)  Generate self-signed certificate (recommended for testing purposes
     only)
 2)  Use an existing certificate located on a Java Keystore (JKS)
 3)  Use an existing certificate located on a PKCS12 keystore
 4)  Use an existing certificate on a PKCS11 token

Enter option [1]: 2

Java Keystore (JKS) path: /ca/ds1-keystore
Keystore PIN: {password}

Truststore options:

 1)  Generate a default JKS truststore
 2)  Use an existing JKS truststore
 3)  Use an existing PKCS12 truststore

Enter option [1]: 2

JKS truststore path: /ca/truststore
Truststore password (can be blank): {password}
```

When using `setup` in non-interactive mode, use the following arguments to configure TLS support.

Argument	Description
`--ldapPort` *<port>*	Indicates that the server should enable support for unencrypted LDAP connections on the specified TCP port. If this argument is not provided, then the server does not accept unencrypted LDAP connections.
`--ldapsPort` *<port>*	Indicates that the server should enable support for LDAPS (LDAP over TLS) on the specified TCP port.
`--httpsPort` *<port>*	Indicates that the server should enable support for HTTPS (for things like SCIM, the Directory REST API, the web-based administration console, etc.) on the specified TCP port.

Argument	Description
`--enableStartTLS`	Indicates that the LDAP connection handler should enable support for the StartTLS extended operation. This argument should only be provided if the `--ldapPort` argument is also given.
`--generateSelfSignedCertificate`	Indicates that setup should generate a self-signed certificate to be presented to clients using LDAPS, HTTPS, and the StartTLS extended operation.
`--useJavaKeyStore` *<path>*	Indicates that the server should use the specified JKS key store to obtain the certificate chain to be presented to clients using LDAPS, HTTPS, and the StartTLS extended operation.
`--usePKCS12KeyStore` *<path>*	Indicates that the server should use the specified PKCS #12 key store to obtain the certificate chain to be presented to clients using LDAPS, HTTPS, and the StartTLS extended operation.
`--usePKCS11KeyStore`	Indicates that the server should use a PKCS #11 key store (e.g., a hardware security module) to obtain the certificate chain to be presented to clients using LDAPS, HTTPS, and the StartTLS extended operation. The JVM must already be configured to access the desired key store via PKCS #11.
`--keyStorePassword` *<password>*	The password needed to interact with the specified JKS, PKCS #12, or PKCS #11 key store. Note that setup assumes that the private key password matches the key store password.
`--keyStorePasswordFile` *<path>*	The path to a file containing the password needed to interact with the specified JKS, PKCS #12, or PKCS #11 key store.
`--certNickname` *<alias>*	The alias of the private key entry in the specified key store that contains the certificate chain to present to clients during TLS negotiation. This argument is optional, but it is recommended if the key store has multiple certificates.
`--useJavaTrustStore` *<path>*	Indicates that the server should use the specified JKS trust store to determine whether to trust any certificate chains that are presented to it during TLS negotiation.
`--usePKCS12TrustStore` *<path>*	Indicates that the server should use the specified PKCS #12 trust store to determine whether to trust any certificate chains that are presented to it during TLS negotiation.

Argument	Description
`--trustStorePassword <password>`	The password needed to interact with the specified JKS or PKCS #11 trust store.
`--trustStorePasswordFile <path>`	The path to a file containing the password needed to interact with the specified JKS or PKCS #11 trust store.
`--rejectInsecureRequests`	Indicates that the server should be configured to reject requests received over insecure connections. This argument can be used in conjunction with the `--ldapPort` argument to allow clients to establish connections that are initially insecure, but requires those connections to be secured with the StartTLS extended operation before they can issue other types of requests.

For example, the following command could be used to set up the server in non-interactive mode with an existing certificate.

```
$ ./setup \
     --no-prompt \
     --acceptLicense \
     --ldapPort 389 \
     --ldapsPort 636 \
     --httpsPort 443 \
     --enableStartTLS \
     --useJavaKeyStore config/keystore \
     --keyStorePasswordFile config/keystore.pin \
     --certNickname server-cert \
     --useJavaTrustStore config/truststore \
     --trustStorePasswordFile config/truststore.pin \
     --baseDN dc=example,dc=com \
     --rootUserDN "cn=Directory Manager" \
     --rootUserPasswordFile root-pw.txt \
     --maxHeapSize 10g \
     --encryptDataWithPassphraseFromFile encryption-settings-password.txt \
     --instanceName ds1 \
     --location Austin \
     --noPropertiesFile


Ping Identity Directory Server 8.2.0.0

Initializing ..... Done
Configuring Directory Server ..... Done
Configuring Certificates ..... Done
Starting Directory Server ..... Done

Access product documentation from docs/index.html
```

Enabling TLS support after setup

If the server has been set up without support for TLS, you can enable it after the fact by obtaining a certificate chain, configuring key and trust manager providers, and configuring connection handlers.

The process for obtaining a certificate has already been discussed in depth. Use `manage-certificates` (or some other tool) to prepare a Java KeyStore (JKS) or PKCS #12 key store with an appropriate certificate chain and private key. You should also create a trust store for use by the server.

Configuring key and trust manager providers
After you have a key store, you can configure a key manager provider to access it.

The server is preconfigured with key manager providers that can be used with Java KeyStore (JKS) or PKCS #12 key stores, named “JKS” and “PKCS12”, respectively. In most cases, the appropriate key manager provider can be updated to reference the key store that you will use.

```
dsconfig set-key-manager-provider-prop \
  --provider-name JKS \
  --set enabled:true \
  --set key-store-file:config/keystore \
  --set key-store-pin-file:config/keystore.pin
```

Use a similar change to configure a trust manager provider to reference the appropriate trust store.

```
dsconfig set-trust-manager-provider-prop \
     --provider-name JKS \
     --set enabled:true \
     --set include-jvm-default-issuers:true \
     --set trust-store-file:config/truststore
     --set trust-store-pin-file:config/truststore.pin
```

Alternatively, if clients and servers are all expected to use certificates signed by issuers included in the JVM’s default trust store, you can simply use the “JVM-Default” trust manager provider.

Configuring connection handlers
After you have configured the key and trust manager providers, you can update the connection handlers to use them.

For the LDAP connection handler, which accepts non-secure connections by default, you can enable StartTLS with a configuration change as in the following example.

```
dsconfig set-connection-handler-prop \
     --handler-name "LDAP Connection Handler" \
     --set allow-start-tls:true \
     --set key-manager-provider:JKS \
     --set trust-manager-provider:JKS \
     --set ssl-cert-nickname:server-cert \
     --set ssl-client-auth-policy:optional
```

If you want to require that clients use StartTLS when connected to the LDAP connection handler, use the reject-insecure-requests global configuration property.

```
dsconfig set-global-configuration-prop \
     --set reject-insecure-requests:true
```

If you did not configure secure communication during setup, then the LDAPS connection handler is disabled. Configuring LDAPS support requires enabling that connection handler and configuring most of the same settings. except `allow-start-tls` must be false and `use-ssl` must be true.

```
dsconfig set-connection-handler-prop \
     --handler-name "LDAPS Connection Handler" \
     --set enabled:true \
     --set key-manager-provider:JKS \
     --set trust-manager-provider:JKS \
     --set ssl-cert-nickname:server-cert \
     --set ssl-client-auth-policy:optional
```

Use a similar configuration change to enable the HTTPS connection handler.

```
dsconfig set-connection-handler-prop \
     --handler-name "HTTPS Connection Handler" \
     --set enabled:true \
     --set listen-port:443 \
     --set key-manager-provider:JKS \
     --set trust-manager-provider:JKS \
     --set ssl-cert-nickname:server-cert
```

Updating the topology registry

After you update the server connection handlers to enable TLS, you should also update the topology registry to provide information about the new configuration.

The topology registry holds information about server instances that are part of the environment and helps facilitate inter-server communication, like replication, mirroring portions of the configuration, and the Directory Proxy Server's automatic backend server discovery functionality.

There are two types of entries that need to be updated: the server instance listener configuration, which provides information needed to trust the TLS certificates presented by instances in the topology, and the server instance configuration, which provides information about options for communicating with those instances.

The server instance listener configuration needs to include the server certificate (that is, the certificate at the head of the chain). This should be the multi-line PEM-formatted representation of the certificate. When using `dsconfig`, this is easiest to import from a file.

```
bin/dsconfig set-server-instance-listener-prop \
  --instance-name ds1 \
  --listener-name ldap-listener-mirrored-config \
  --set server-ldap-port:636 \
  --set connection-security:ssl \
  --set 'listener-certificate</ca/ds1-cert.pem'
```

ⓘ **Note:**

The use of the less-than operator in the last line indicates that the value should be read from a file rather than provided directly. Also, the property name and path might need to be enclosed in single straight quotes to prevent the shell from interpreting the less-than symbol as an attempt to redirect input.

The server instance configuration object should also be updated to reflect the new methods that are available to communicate with that instance, and the preferred-security property indicates what mechanism other instances in the topology should try to use when communicating with that instance. The following example demonstrates setting the LDAPS and HTTPS ports to indicate that StartTLS support has been enabled and to indicate that other instances should use SSL (LDAPS) when communicating with this instance.

```
dsconfig set-server-instance-prop \
     --instance-name ds1 \
     --set ldaps-port:636 \
     --set https-port:443 \
     --set preferred-security:ssl \
     --set start-tls-enabled:true
```

Configuring supported TLS protocols and cipher suites

By default, the PingDirectory Server enables support for the following TLS protocol versions:

- TLS 1

- TLS 1.1
- TLS 1.2
- TLS 1.3 (if supported by the underlying Java Virtual Machine (JVM))

This attempts to strike a good balance between providing the best security and maintaining compatibility with legacy clients.

Modern security best practices recommend only enabling support for TLS 1.2 and TLS 1.3, but some legacy LDAP clients may only support older versions. However, if you are confident that your LDAP clients support the more modern protocols, you can configure the server to only support those versions.

The server also tries to select an appropriate set of cipher suites to use for the TLS communication. It excludes suites that use known-weak key exchange, encryption, and digest algorithms and prioritizes key exchange algorithms that support forward secrecy over those that do not. However, as with the TLS protocol you can also explicitly customize the set of cipher suites that you wish to support.

The set of TLS protocols and cipher suites can be customized on a per-connection-handler basis using the connection handler's `ssl-protocol` and `ssl-cipher-suite` configuration properties. You should also customize those properties in the crypto manager configuration, as the server uses that for other purposes like securing replication traffic.

```
dsconfig set-crypto-manager-prop \
     --set ssl-protocol:TLSv1.2 \
     --set ssl-protocol:TLSv1.3 \
     --set ssl-cipher-suite:TLS_AES_256_GCM_SHA384 \
     --set ssl-cipher-suite:TLS_AES_128_GCM_SHA256 \
     --set ssl-cipher-suite:TLS_ECDHE_ECDSA_WITH_AES_256_GCM_SHA384 \
     --set ssl-cipher-suite:TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384 \
     --set ssl-cipher-suite:TLS_ECDHE_ECDSA_WITH_AES_128_GCM_SHA256 \
     --set ssl-cipher-suite:TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256 \
     --set ssl-cipher-suite:TLS_ECDHE_ECDSA_WITH_AES_256_CBC_SHA384 \
     --set ssl-cipher-suite:TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA384 \
     --set ssl-cipher-suite:TLS_ECDHE_ECDSA_WITH_AES_256_CBC_SHA \
     --set ssl-cipher-suite:TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA \
     --set ssl-cipher-suite:TLS_ECDHE_ECDSA_WITH_AES_128_CBC_SHA256 \
     --set ssl-cipher-suite:TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA256 \
     --set ssl-cipher-suite:TLS_ECDHE_ECDSA_WITH_AES_128_CBC_SHA \
     --set ssl-cipher-suite:TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA \
     --set ssl-cipher-suite:TLS_EMPTY_RENEGOTIATION_INFO_SCSV

dsconfig set-connection-handler-prop \
     --handler-name "LDAPS Connection Handler" \
     --set ssl-protocol:TLSv1.2 \
     --set ssl-protocol:TLSv1.3 \
     --set ssl-cipher-suite:TLS_AES_256_GCM_SHA384 \
     --set ssl-cipher-suite:TLS_AES_128_GCM_SHA256 \
     --set ssl-cipher-suite:TLS_ECDHE_ECDSA_WITH_AES_256_GCM_SHA384 \
     --set ssl-cipher-suite:TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384 \
     --set ssl-cipher-suite:TLS_ECDHE_ECDSA_WITH_AES_128_GCM_SHA256 \
     --set ssl-cipher-suite:TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256 \
     --set ssl-cipher-suite:TLS_ECDHE_ECDSA_WITH_AES_256_CBC_SHA384 \
     --set ssl-cipher-suite:TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA384 \
     --set ssl-cipher-suite:TLS_ECDHE_ECDSA_WITH_AES_256_CBC_SHA \
     --set ssl-cipher-suite:TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA \
     --set ssl-cipher-suite:TLS_ECDHE_ECDSA_WITH_AES_128_CBC_SHA256 \
     --set ssl-cipher-suite:TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA256 \
     --set ssl-cipher-suite:TLS_ECDHE_ECDSA_WITH_AES_128_CBC_SHA \
     --set ssl-cipher-suite:TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA \
     --set ssl-cipher-suite:TLS_EMPTY_RENEGOTIATION_INFO_SCSV

dsconfig set-connection-handler-prop \
     --handler-name "LDAP Connection Handler" \
     --set ssl-protocol:TLSv1.2 \
```

```
     --set ssl-protocol:TLSv1.3 \
     --set ssl-cipher-suite:TLS_AES_256_GCM_SHA384 \
     --set ssl-cipher-suite:TLS_AES_128_GCM_SHA256 \
     --set ssl-cipher-suite:TLS_ECDHE_ECDSA_WITH_AES_256_GCM_SHA384 \
     --set ssl-cipher-suite:TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384 \
     --set ssl-cipher-suite:TLS_ECDHE_ECDSA_WITH_AES_128_GCM_SHA256 \
     --set ssl-cipher-suite:TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256 \
     --set ssl-cipher-suite:TLS_ECDHE_ECDSA_WITH_AES_256_CBC_SHA384 \
     --set ssl-cipher-suite:TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA384 \
     --set ssl-cipher-suite:TLS_ECDHE_ECDSA_WITH_AES_256_CBC_SHA \
     --set ssl-cipher-suite:TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA \
     --set ssl-cipher-suite:TLS_ECDHE_ECDSA_WITH_AES_128_CBC_SHA256 \
     --set ssl-cipher-suite:TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA256 \
     --set ssl-cipher-suite:TLS_ECDHE_ECDSA_WITH_AES_128_CBC_SHA \
     --set ssl-cipher-suite:TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA \
     --set ssl-cipher-suite:TLS_EMPTY_RENEGOTIATION_INFO_SCSV

dsconfig set-connection-handler-prop \
     --handler-name "HTTPS Connection Handler" \
     --set ssl-protocol:TLSv1.2 \
     --set ssl-protocol:TLSv1.3 \
     --set ssl-cipher-suite:TLS_AES_256_GCM_SHA384 \
     --set ssl-cipher-suite:TLS_AES_128_GCM_SHA256 \
     --set ssl-cipher-suite:TLS_ECDHE_ECDSA_WITH_AES_256_GCM_SHA384 \
     --set ssl-cipher-suite:TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384 \
     --set ssl-cipher-suite:TLS_ECDHE_ECDSA_WITH_AES_128_GCM_SHA256 \
     --set ssl-cipher-suite:TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256 \
     --set ssl-cipher-suite:TLS_ECDHE_ECDSA_WITH_AES_256_CBC_SHA384 \
     --set ssl-cipher-suite:TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA384 \
     --set ssl-cipher-suite:TLS_ECDHE_ECDSA_WITH_AES_256_CBC_SHA \
     --set ssl-cipher-suite:TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA \
     --set ssl-cipher-suite:TLS_ECDHE_ECDSA_WITH_AES_128_CBC_SHA256 \
     --set ssl-cipher-suite:TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA256 \
     --set ssl-cipher-suite:TLS_ECDHE_ECDSA_WITH_AES_128_CBC_SHA \
     --set ssl-cipher-suite:TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA \
     --set ssl-cipher-suite:TLS_EMPTY_RENEGOTIATION_INFO_SCSV
```

See the `configure-enabled-tls-protocols.dsconfig` and `configure-enabled-tls-cipher-suites.dsconfig` files in the `config/sample-dsconfig-batch-files` directory for more information.

Using TLS in command-line tools

The PingDirectory software includes a wide variety of command-line tools that can help manage and interact with the server.

Many of these tools communicate with the server over LDAP, and it is important to understand how to secure that communication with TLS.

Common arguments for TLS communication

The most useful TLS-related arguments our LDAP tools offer include the following.

Argument	Description
`--hostname` *<address>*	The address of the server to which the connection should be established.

Argument	Description
`--port <port>`	The TCP port of the server to which the connection should be established. 389 is the standard port for non-secure LDAP (or LDAP to be secured with StartTLS) and 636 is the standard port for secure LDAPS, but many deployments use alternate ports (especially non-privileged ports above 1024).
`--useSSL`	Indicates that the tool should establish an LDAPS connection that is secured with TLS.
`--useStartTLS`	Indicates that the tool should establish an initially insecure LDAP connection that is then secured with the StartTLS extended operation.
`--trustStorePath <path>`	The path to a trust store that should be used in the course of determining whether to trust the certificate chain presented by the server during TLS negotiation. If neither this argument nor the `--trustAll` argument is provided, then the tool interactively prompts about whether to trust the server certificate if it is not signed by an issuer in the Java Virtual Machine (JVM's) default trust store.
`--trustStoreFormat <format>`	The format for the trust store. This is typically be Java KeyStore (JKS) or PKCS12.
`--trustStorePassword <password>`	The password needed to access the contents of the trust store.
`--trustStorePasswordFile <path>`	The path to a file containing the password needed to access the contents of the trust store.
`--trustAll`	Indicates that the tool should blindly trust any TLS certificate chain that is presented to it. This is not recommended for general use, but it can be useful for troubleshooting purposes.
`--keyStorePath <path>`	The path to a key store that should be used if a client certificate chain should be presented to the server. This should typically be omitted unless the server has been configured to require clients to present a certificate or unless you want to use the certificate to authenticate using SASL EXTERNAL.
`--keyStoreFormat <format>`	The format for the key store. It is typically JKS or PKCS12.
`--keyStorePassword <password>`	The password needed to access the key store.
`--keyStorePasswordFile <path>`	The path to a file containing the password needed to access the key store.

Argument	Description
`--certNickname <alias>`	The alias of the private key entry in the key store that should be used to obtain the certificate chain to present to the server.
`--useSASLExternal`	Indicates that the client should authenticate using the EXTERNAL SASL mechanism, which typically identifies the client using the certificate chain that was presented during TLS negotiation.
`--enableSSLDebugging`	Indicates that the tool should enable low-level TLS debugging provided by the JVM.

The easiest way to test TLS communication with the Directory Server is to use a command like ldapsearch.

```
$ bin/ldapsearch \
  --hostname ds1.example.com \
  --port 636 \
  --useSSL \
        --trustStorePath config/truststore \
  --baseDN "" \
  --scope base \
  "(objectClass=*)"
dn:
objectClass: top
objectClass: ds-root-dse
startupUUID: 8d574122-4584-4522-96d9-0cdcb9d2e339
startTime: 20191113061149Z

# Result Code:  0 (success)
# Number of Entries Returned:  1
```

If you don't provide any trust-related arguments, then the tool automatically checks each of the following to determine whether it can trust the certificate chain that the server presented:

- The `config/truststore` file for the local instance
- The certificates defined in the local instance's topology registry
- The JVM's default trust store

If the presented certificate chain can be trusted through one of those mechanisms, then the tool establishes the connection without the need for any further interaction. However, if the certificate chain is still not trusted after checking each of those locations, then the tool displays information about that certificate chain and interactively prompts to determine if it should be trusted.

If you want to ensure that the tool does not prompt about whether to trust the certificate chain, then provide trust-related arguments like `--trustStorePath`. If the certificate chain cannot be trusted using the information in the provided trust store, then the tool exits with an error.

If you want to present a client certificate chain to the server (for example, because the server's connection handler has been configured with an `ssl-client-auth-policy` value of required, or because you want to use the certificate to authenticate using the SASL EXTERNAL mechanism), then you must provide at least the key store and its corresponding password. You can also specify the alias of the certificate chain to present, which you should probably do if your client key store contains multiple certificates.

```
$ bin/ldapsearch \
     --hostname ds1.example.com \
     --port 636 \
     --useSSL \
     --trustStorePath config/truststore.p12 \
```

```
     --trustStorePasswordFile config/truststore.pin \
     --trustStoreFormat PKCS12 \
     --keyStorePath client-keystore \
     --keyStorePasswordFile client-keystore.pin \
     --certNickname client-cert \
     --saslOption mech=EXTERNAL \
     --baseDN "" \
     --scope base \
     "(objectClass=*)"
dn:
objectClass: top
objectClass: ds-root-dse
startupUUID: c8724159-8c37-45eb-b210-879bfcf74ad6
startTime: 20191113154023Z

# Result Code:  0 (success)
# Number of Entries Returned:  1
```

If at any point you encounter a TLS-related problem that you can't figure out by looking at the `ldapsearch` output or server logs, then you can try using the `--enableSSLDebugging` option. This enables the JVM's support for low-level debugging of TLS processing, as described in the next section.

Troubleshooting TLS-related problems

Ideally, the TLS configuration functions properly and all clients can communicate securely with the Directory Server. If errors occur, there are several things you can do to help troubleshoot the problem.

Log Messages

The first place to check is the server's access log.

If the client can successfully establish a TCP connection to the server, which must happen before TLS negotiation can begin, the access log should show a `CONNECT` message with the source and destination address and port for the connection, the protocol, and the selected client connection policy. If you don't see that `CONNECT` message, then that suggests that the client might not be able to communicate at all with the server. That might mean there is a network problem, that a firewall is blocking the communication, or something as simple as the client is trying to use the wrong address or port.

If you see the `CONNECT` message in the access log, then that message should include a `conn` element that provides the connection ID for that connection. You can then use the `search-logs` tool to see if there are any other log messages for that client connection. For example, if the connection ID is 12345, then the following command can show the complete set of log messages for that connection.

```
$ bin/search-logs --logFile logs/access conn=12345
```

When trying to use LDAPS, the next log message that you see should either be a `SECURITY-NEGOTIATION` or a `DISCONNECT` message. `SECURITY-NEGOTIATION` messages should indicate that the client and server successfully completed the negotiation process and should include details about that negotiation, like the TLS protocol and selected cipher suite. If you see a `SECURITY-NEGOTIATION` message, then it indicates that the problem likely happened after the TLS session was established.

On the other hand, if the `CONNECT` message is immediately followed by a `DISCONNECT` message, then that suggests that the problem is likely a failure in the TLS negotiation process. In such cases, the message should include a `reason` element that provides more information about the reason for the disconnect. If the failure occurred during TLS negotiation, then how useful that message is depends at least in part on whether the failure occurred on the client or the server. If the server decided to abort the negotiation, then the message ideally contains the exact reason. If the problem occurred on the client, then the log message likely only contains the general category for the failure. That is because the TLS protocol does not provide a mechanism for conveying detailed error messages but only offers a basic alert

mechanism with a fixed set of alert types. For example, if the problem is that the client did not trust the certificate chain that the server presented to it, the client might know exactly why it rejected the chain, but the server probably only gets an alert like `certificate_unknown`. In such cases, you might need to see if the client provides any more detail about the problem.

If the access log does not appear to provide any useful information, then you might also want to check the server error log. The error log won't normally include information about problems related to client communication, but it can offer useful information in certain circumstances, such as if an internal error within the server is interfering with the communication attempt.

manage-certificates check-certificate-usability

The `manage-certificates` tool offers a `check-certificate-usability` subcommand that can be used to examine a specified entry in a key store and identify any potential problems with it that might interfere with secure communication.

Some of the things it checks include:

- It makes sure the specified entry in the key store includes a private key and a complete certificate chain.
- It checks whether the certificate at the root of the chain is found in the JVM's default set of trusted certificates.
- It makes sure that the current time is within the validity window for all of the certificates in the chain.
- It validates the signatures for all certificates in the chain.
- It warns if the end-entity certificate is self-signed.
- It warns if the end-entity certificate does not contain an extended key usage extension with the "serverAuth" usage.
- It warns if any of the issuer certificates does not have a key usage extension with the "keyCertSign" usage.
- It warns if any of the issuer certificates does not have a basic constraints extension indicating that it can operate as a certification authority. It reports an error if the chain violates a path length constraint.
- It ensures that the signature algorithm uses a strong message digest algorithm, like SHA-256. It reports an error for weak digest algorithms like MD5 or SHA-1, and a warning for unrecognized digest algorithms.
- It ensures that none of the certificates using an RSA key pair have a key size less than 2048 bits.

The following example demonstrates the usage for this command and its output when no problems are identified.

```
$ bin/manage-certificates check-certificate-usability \
     --keystore config/keystore \
     --keystore-password-file config/keystore.pin \
     --alias server-cert

Successfully retrieved the certificate chain for alias 'server-cert':

Subject DN:  CN=ds1.example.com,O=Example Corp,C=US
Issuer DN:  CN=Example Intermediate CA,O=Example Corp,C=US
Validity Start Time:  Tuesday, November 12, 2019 at 03:52:44 PM CST (5
 minutes, 45 seconds ago)
Validity End Time:  Wednesday, November 11, 2020 at 03:52:44 PM CST (364
 days, 23 hours, 54 minutes, 14 seconds from now)
Validity State:  The certificate is currently within the validity window.
Signature Algorithm:  SHA-256 with RSA
Public Key Algorithm:  RSA (2048-bit)
SHA-1 Fingerprint:
 84:e4:00:b9:f0:6b:58:bb:ac:67:79:28:2f:43:9f:e3:ac:24:ee:98
SHA-256 Fingerprint:
 63:85:4d:2c:50:ea:a8:84:54:e0:73:9a:e7:5b:e7:1b:06:85:0e:28:2b:76:a9:8b:57:fc:27:f7:60
```

```
Subject DN:  CN=Example Intermediate CA,O=Example Corp,C=US
Issuer DN:  CN=Example Root CA,O=Example Corp,C=US
Validity Start Time:  Tuesday, November 12, 2019 at 03:52:42 PM CST (5
 minutes, 47 seconds ago)
Validity End Time:  Monday, November 7, 2039 at 03:52:42 PM CST (7299 days,
 23 hours, 54 minutes, 12 seconds from now)
Validity State:  The certificate is currently within the validity window.
Signature Algorithm:  SHA-256 with RSA
Public Key Algorithm:  RSA (4096-bit)
SHA-1 Fingerprint:
 de:da:3d:fc:d4:1f:67:79:0a:a1:5a:cd:ca:4a:7e:a5:d3:46:88:27
SHA-256 Fingerprint:
 02:3c:af:ad:b7:07:81:89:45:48:d0:09:31:a8:90:c4:17:11:1c:00:11:fd:49:b2:2c:ba:ac:dd:c4

Subject DN:  CN=Example Root CA,O=Example Corp,C=US
Issuer DN:  CN=Example Root CA,O=Example Corp,C=US
Validity Start Time:  Tuesday, November 12, 2019 at 03:52:38 PM CST (5
 minutes, 51 seconds ago)
Validity End Time:  Monday, November 7, 2039 at 03:52:38 PM CST (7299 days,
 23 hours, 54 minutes, 8 seconds from now)
Validity State:  The certificate is currently within the validity window.
Signature Algorithm:  SHA-256 with RSA
Public Key Algorithm:  RSA (4096-bit)
SHA-1 Fingerprint:
 8e:03:e4:58:e6:e3:59:9a:55:77:c0:88:3c:fa:d7:29:f4:ff:de:6c
SHA-256 Fingerprint:
 95:54:0d:e2:aa:48:29:c1:25:7c:20:69:c0:27:33:31:81:07:02:2e:00:24:ae:49:5e:98:bd:a3:72

OK:  The certificate chain is complete.  Each subsequent certificate is
the issuer for the previous certificate in the chain, and the chain ends
with a self-signed certificate.

OK:  Certificate 'CN=ds1.example.com,O=Example Corp,C=US' has a valid
signature.

OK:  Certificate 'CN=Example Intermediate CA,O=Example Corp,C=US' has a
valid signature.

OK:  Certificate 'CN=Example Root CA,O=Example Corp,C=US' has a valid
signature.

OK:  Certificate 'CN=ds1.example.com,O=Example Corp,C=US' will expire at
Wednesday, November 11, 2020 at 03:52:44 PM CST (364 days, 23 hours, 54
minutes, 14 seconds from now), which is not in the near future.

OK:  Issuer certificate 'CN=Example Intermediate CA,O=Example Corp,C=US'
will expire at Monday, November 7, 2039 at 03:52:42 PM CST (7299 days, 23
hours, 54 minutes, 12 seconds from now), which is not in the near future.

OK:  Issuer certificate 'CN=Example Root CA,O=Example Corp,C=US' will
expire at Monday, November 7, 2039 at 03:52:38 PM CST (7299 days, 23
hours, 54 minutes, 8 seconds from now), which is not in the near future.

OK:  Certificate 'CN=ds1.example.com,O=Example Corp,C=US' at the head of
the chain includes an extended key usage extension, and that extension
includes the serverAuth usage.

OK:  Issuer certificate 'CN=Example Intermediate CA,O=Example Corp,C=US'
includes a basic constraints extension, and the certificate chain
satisfies those constraints.

OK:  Issuer certificate 'CN=Example Intermediate CA,O=Example Corp,C=US'
includes a key usage extension with the keyCertSign usage flag set to
true.
```

```
OK:  Issuer certificate 'CN=Example Root CA,O=Example Corp,C=US' includes
a basic constraints extension, and the certificate chain satisfies those
constraints.

OK:  Issuer certificate 'CN=Example Root CA,O=Example Corp,C=US' includes
a key usage extension with the keyCertSign usage flag set to true.

OK:  Certificate 'CN=ds1.example.com,O=Example Corp,C=US' uses a signature
algorithm of 'SHA-256 with RSA', which is is considered strong.

OK:  Certificate 'CN=Example Intermediate CA,O=Example Corp,C=US' uses a
signature algorithm of 'SHA-256 with RSA', which is is considered strong.

OK:  Certificate 'CN=Example Root CA,O=Example Corp,C=US' uses a signature
algorithm of 'SHA-256 with RSA', which is is considered strong.

OK:  Certificate 'CN=ds1.example.com,O=Example Corp,C=US' has a 2048-bit
RSA public key, which is considered strong.

OK:  Certificate 'CN=Example Intermediate CA,O=Example Corp,C=US' has a
4096-bit RSA public key, which is considered strong.

OK:  Certificate 'CN=Example Root CA,O=Example Corp,C=US' has a 4096-bit
RSA public key, which is considered strong.

No usability errors or warnings were identified while validating the
certificate chain.
```

If any usability issues are identified, they might be responsible for the communication problems.

Low-level TLS debugging

If nothing else helps, then you might need to resort to low-level debugging options.

In this case, the best option is probably to enable the Java Virtual Machine (JVM's) support for TLS debugging. Many of the command-line tools provided with the Directory Server, like ldapsearch, offer an `--enableSSLDebugging` argument that can simplify this process, but for other tools, you can edit the `config/java.properties` file to add `-Djavax.net.debug=all` to the set of properties for the desired tool and then run the `bin/dsjavaproperties` command to make the change take effect. The next time you run the tool, you will get voluminous output detailing all of the TLS-related processing that the JVM is performing. Something in that output should allow you or our support personnel to identify the issue.

This low-level debugging is also available within the server by adding `-Djavax.net.debug=all` to the `start-server.java-args` property in the `config/java.properties` file, running `bin/dsjavaproperties`, and restarting the server. Because this requires a server restart (and then another one when you want to turn off the debugging), it is not a very attractive option. You might be able to obtain more information about the problem without a restart by using the debugging support built into the server. You can do that with the following configuration changes.

```
dsconfig create-debug-target \
     --publisher-name "File-Based Debug Logger" \
     --target-name
 com.unboundid.directory.server.extensions.TLSConnectionSecurityProvider \
     --set debug-level:verbose

dsconfig set-log-publisher-prop \
     --publisher-name "File-Based Debug Logger" \
     --set enabled:true
```

After these changes, the `logs/debug` file captures a substantial amount of information about the TLS-related processing that the server is performing. It won't have quite as much detail as the debugging information built into the JVM, but it can still allow pinpointing the cause of the problem and identifying the solution. When this debugging is no longer needed, the debug log publisher can be disabled, and the debug target can be removed.

```
dsconfig set-log-publisher-prop \
     --publisher-name "File-Based Debug Logger" \
     --set enabled:false

dsconfig delete-debug-target \
     --publisher-name "File-Based Debug Logger" \
     --target-name
 com.unboundid.directory.server.extensions.TLSConnectionSecurityProvider
```

If you need to troubleshoot TLS communication with a non-Java client that doesn't offer its own TLS debugging mechanism, and if the server-side debugging support is not sufficient, then the best remaining option can be to use a network protocol analyzer to capture the communication between the client and the server and examine its content. The free and open source Wireshark utility, available at *https://www.wireshark.org/*, is an excellent graphical tool that runs on a variety of platforms and provides excellent support for understanding TLS communication. Even if you're unable to decipher the encrypted content, you can see at least some of the handshake messages. Unfortunately, when using TLS version 1.3, more of the handshake is now encrypted than with earlier versions of the protocol. This is terrific for security and privacy but unfortunately can interfere with troubleshooting attempts.

Additional mechanisms for securing communication

TLS provides a strong way to ensure that unauthorized observers aren't able to interpret the network communication to and from the PingDirectory Server.

It also includes a trust mechanism to help clients ensure that they are communicating with a legitimate server and not an impostor. However, there are additional steps you can take to secure network communication.

Secure name service configuration

If possible, you should use a secure name service like DNS over TLS or DNS over HTTPS. Regular DNS, especially DNS over UDP connections, is vulnerable to hijacking attacks.

If an attacker is able to run their own DNS server, and if that server is able to respond more quickly than the legitimate DNS server, then clients can be tricked into establishing connections to the wrong server.

If a secure DNS option is not available, then another option can be to use host files for name resolution. However, this option can be difficult to maintain in dynamic environments in which server addresses might change. It is also not a feasible option if you do not have control over the client systems.

Name service caching

If name resolution is slow, then it can adversely affect server performance.

If the server is unable to resolve a host name to the corresponding address, then it might be unable to establish a connection to an external system. In some cases, it can also affect the ability to accept client connections or evaluate access control rules.

The server logs a message if attempts to resolve a host name to an IP address fail or take a long time to complete. This can help make it easier to diagnose problems related to name resolution, but it would be better to prevent those problems in the first place.

The JVM provides its own address caching facility that can help with this. It maintains its own internal cache that maps host names to IP addresses, and each mapping is associated with a Time To Live (TTL)

value that indicates how long it should be used. If the mapping between host names and IP addresses is stable in your environment, then you might want to configure the JVM to use a large TTL value to reduce its dependency on the underlying name service. From a security perspective, this is primarily useful for cases in which you cannot rely on a secure name service or host file, but it can also help mitigate the possibility of problems that could arise in the event of a name service outage. You can use the `network-address-cache-ttl` property in the global configuration to tune this value.

You might also want to consider running a caching name server on the same system as the server to provide an additional layer of protection against name service outages and to reduce network latency for name service requests.

Strong TCP sequence numbers

Use strong TCP sequence numbers to prevent already-established TCP connections from being hijacked.

Reject source-routed packets

Source-routed packets allow a packet's sender to specify which route it should take to its destination.

An attacker might be able to use this capability to trick the client into communicating with the wrong system. Source-routed packets are rarely used for legitimate communication.

Reject ICMP redirects

The Internet Control Message Protocol (ICMP) offers features that can ensure that traffic gets from its source to its destination as efficiently as possible, but it can also help attackers hijack existing sessions.

ICMP redirects are intended to provide a mechanism for a router to let a client know about a better way to reach the target system, but they are rarely needed in private networks, and attackers can use them to trick the client into communicating with the wrong system.

Encrypt all inter-system communication

Any communication that the PingDirectory Server itself has with other systems should be encrypted to resist observation and interception.

This also applies to any communication that the underlying system needs to perform, including name service resolution, use of network-based filesystems, remote logging, shell access, and file transfer.

Restricting client access

One way to prevent clients from gaining unauthorized access to the data is to prevent unauthorized clients from establishing connections or to terminate connections if conditions change in a way that the server determines is no longer acceptable.

Restricting access through network access controls

One common and effective way to prevent connections from unauthorized clients is to prevent those clients from reaching the server.

This can be accomplished using firewall software on the underlying system or through access controls in network hardware like firewalls and routers.

Restricting access through connection handlers

For connection attempts that are able to reach the PingDirectory Server, the server itself can make decisions about whether those connections should be accepted.

The first layer of defense is at the connection handler that accepts the connection. Connection handlers offer the following configuration properties for determining which clients should be accepted and which should be rejected:

allowed-client

An optional set of address masks that indicate which clients are allowed to establish connections to that connection handler. If one or more allowed-client values are defined, then only clients whose address matches one of those patterns are permitted.

denied-client

An optional set of address masks that indicate which clients are not allowed to establish connections to that connection handler. If one or more denied-client values are defined, then any connection from a client whose address matches one of those patterns are terminated.

Any values provided for the `allowed-client` and `denied-client` properties should be formatted as address masks. These address masks can take several forms, including:

- They can be raw IPv4 addresses, like `1.2.3.4`.
- They can be raw IPv6 addresses. These addresses can use the full hexadecimal representation, such as `2001:fecd:ba23:cd1f:dcb1:1010:9234:4088` optionally surrounded by square brackets. They can also use the shorthand notation when appropriate, such as `::1` and IPv6 representations of IPv4 addresses can end with the dotted IPv4 representation, such as `0:0:0:0:0:ffff:1.2.3.4`.
- They can be IPv4 addresses that use the asterisk as a wildcard character in one or more of the octets, such as `1.2.3.*` or `*.*.*.*`.
- They can be an IPv4 or IPv6 address using CIDR notation to indicate the number of bits that are required to match, such as `1.2.3.0/24` or `::1/128`.
- They can be IPv4 addresses followed by a slash and a subnet mask, such as `1.2.3.4/255.255.255.0`.
- They can use resolvable host names, whether complete or using asterisks as wildcards, such as `client.example.com` or `*.example.com`.

For example, to configure the LDAP connection handler so that it only accepts client connections from the 192.168.0.0/24 subnet, you can use a change as in the following example.

```
dsconfig set-connection-handler-prop \
     --handler-name "LDAP Connection Handler" \
     --set allowed-client:192.168.0.0/24
```

Using an `allowed-client` value of either `192.168.0.0/255.255.255.0` or `192.168.0.*` would also achieve the same result since they are equivalent ways to express the same range of client addresses.

Restricting access through client connection policies

Whenever a client establishes a connection to the PingDirectory server, that connection is associated with a client connection policy, which can restrict the kinds of requests that the client can issue and impose resource limits for that connection.

The server uses the following properties to determine which client connection policy should be associated with a client connection:

evaluation-order-index

An integer value that specifies the order in which the client connection policy will be evaluated relative to other policies. Each client connection policy must have a unique evaluation-order-index value.

connection-criteria

An optional set of criteria that indicates which connections are allowed to be associated with the client connection policy. That criteria can take several things into account, including the address of the client, the connection handler that accepted the connection, whether it is communicating with the server over a secure connection, whether the client is authenticated, the authentication

mechanism, the location or content of the authenticated user's entry, the groups in which that user is a member, and the set of privileges that they have. If the client connection policy is not associated with any connection criteria, then it matches any connection. See the *Connection criteria* section of this document for more information.

`terminate-connection`

A Boolean value that indicates whether to terminate any client connection in which the client connection policy is the first one to match the client connection.

Whenever the server accepts a connection, it iterates through all enabled client connection policies in order from lowest `evaluation-order-index` value to highest. The first policy that the server encounters that either does not have connection criteria or that has connection criteria that matches the client connection is assigned to that connection. If the connection cannot be associated with any client connection policy because all enabled policies have criteria that do not match the client connection, or if the first matching policy has a `terminate-connection` value of true, then the connection is terminated.

After processing each bind operation (which might change the authentication state for the client connection) as well as after each StartTLS extended operation (which might change the communication security for the connection) the server re-selects the client connection policy to use for that connection. It might assign the same policy or a different policy to that connection, or it might terminate the connection.

Restricting access through operational attributes in user entries

The PingDirectory Server also defines a number of operational attributes that can be placed in user entries to indicate the context in which their account can be used.

These attributes include the following.

Attribute	Description
`ds-auth-allowed-address`	Can be used to provide a set of address masks in the same format used by the allowed-client property in the connection handler configuration to indicate which clients are allowed to authenticate as the user. If any allowed addresses are defined and a client attempts to authenticate as the user from a client whose address does not match one of these patterns, then the bind attempt is rejected.
`ds-auth-allowed-authentication-type`	Can be used to restrict the ways in which the user can authenticate to the server. Values can be either `simple` to indicate that the user can authenticate with LDAP simple authentication or "`sasl <mechanism>`" such as "`sasl EXTERNAL`" to indicate that the user can authenticate with the specified SASL mechanism. If any allowed authentication types are defined and a client attempts to authenticate using a mechanism that is not included in this list, then the bind attempt is rejected.

Attribute	Description
`ds-auth-require-secure-authentication`	Can be used to indicate whether the user is required to authenticate to the server in a secure manner that does not reveal the credentials to a network observer whether by authenticating over a secure connection or by using an authentication mechanism that protects the credentials in transit. If this is set to true and a client attempts to authenticate as the user in an insecure manner, then the bind attempt is rejected.
`ds-auth-require-secure-connection`	Can be used to indicate whether the user is required to communicate with the server over an encrypted connection. While this is similar to the `ds-auth-require-secure-authentication` property, if it is set to true, then the user is only allowed to authenticate over a secure connection: it will not allow the client to authenticate over an insecure connection even if the authentication mechanism does not reveal the user's credentials to an external observer.
`ds-auth-is-proxyable`	Indicates whether the user's account can be used as an alternate authorization identity, such as using the proxied authorization request control, or as the authorization identity of a SASL bind. Values of this attribute can be one of the following: **`allowed`** Indicates that the account can optionally be used as an alternate authorization identity. This is the default behavior used for accounts that do not include the `ds-auth-is-proxyable` attribute. **`prohibited`** Indicates that the account cannot be used as an alternate authorization identity. Operations can only be processed as this user by clients that have directly authenticated as that user. **`required`** Indicates that the account can only be used as an alternate authorization identity and is not allowed to directly authenticate to the server.
`ds-auth-is-proxyable-by`	The distinguished names (DNs) of the users that are allowed to request this account as an alternate authorization identity. If one or more ds-auth-is-proxyable-by values are configured, then any attempt to proxy as the user from some account whose DN is not listed is rejected.

Attribute	Description
`ds-auth-is-proxyable-by-group`	The DNs of the groups whose members are allowed to request this account as an alternate authorization identity. If one or more group DNs are provided, then any attempt to proxy as the user from an account that is not a member of any of those groups is rejected.
`ds-auth-is-proxyable-by-url`	A set of LDAP URLs that can be used to identify users that will be allowed to request this account as an alternate authorization identity. If one or more LDAP URLs are provided, then any attempt to proxy as the user from an account that does not match the criteria represented by any of those URLs is rejected.
`ds-auth-may-proxy-as`	The DNs of the accounts that the user can request as an alternate authorization identity. If one or more `ds-auth-may-proxy-as` values are provided and the client attempts to proxy as any user whose DN is not listed, then that attempt is rejected.
`ds-auth-may-proxy-as-group`	The DNs of the groups whose members can be used as alternate authorization identities by the user. If one or more group DNs are provided and the user attempts to proxy as a user that is not a member of any of those groups, then that attempt is rejected.
`ds-auth-may-proxy-as-url`	A set of LDAP URLs that can be used to identify accounts that the user can request as an alternate authorization identity. If one or more LDAP URLs are provided, then an attempt to proxy as an account whose entry does not match the criteria from any of those LDAP URLs is rejected.

These operational attributes can be set as real or virtual attributes in the target user's entry.

Restricting access with plugins

The UnboundID Server SDK provides support for creating a wide range of extensions that can be configured to run in the server. One of the available extension types is `Plugin`, which can be used to invoke custom code at various points in server processing.

If a custom plugin is configured in the server with a `postconnect` plugin type, then its `doPostConnect` method is invoked for any new connection that is established. This method can examine information that the server has about the client connection, and if it determines that the connection should not be allowed, then it can terminate the connection.

Plugins can also intercept operation requests and responses to invoke custom processing for them. If a client requests an operation that a plugin decides should not be allowed, then the plugin can reject the operation or terminate the connection.

Lockdown mode

The PingDirectory Server offers a lockdown mode in which it reports itself as unavailable and only allows requests from clients with the `lockdown-mode` privilege.

Lockdown mode provides a way for the server to be online so that administrators can investigate a problem or perform some disruptive administrative action, but in a manner that causes it to be unavailable to most clients.

The PingDirectory Server can automatically place itself in lockdown mode under certain circumstances. Some of these include:

- If the access control handler encounters a malformed access control rule on startup. The server does its best to prevent invalid access control rules from being created, but if one does make it through, the server enters lockdown mode rather than running with a potentially incomplete access control policy.
- If an unrecoverable error occurs while interacting with a backend database based on the `unrecoverable-database-error-mode` global configuration property.
- If the server is missing replication changes that are no longer available in the replication database based on the `lockdown-on-missed-replication-changes` global configuration properly.
- If available disk space gets too low, as determined by the disk space usage monitor provider's low-space-error-size-threshold and low-space-error-percent-threshold properties.
- If an error occurs while attempting to log a message based on the `logging-error-behavior` property in the log publisher configuration.

The server can also be placed in lockdown mode at any time using the `enter-lockdown-mode` command-line tool, or the enter lockdown mode administrative task that the tool uses behind the scenes. The start-server command also provides a `--lockdownMode` argument that can be used to make the server enter lockdown mode before startup completes.

Once the server enters lockdown mode, that mode stays in effect until the server is restarted or until the `leave-lockdown-mode` command or the underlying administrative task is used. Lockdown mode does not persist across server restarts unless it is automatically triggered by a condition that still exists after the restart.

Criteria

The PingDirectory Server provides a powerful criteria subsystem that allows it to match connections, requests, responses, entries, and references based on a wide variety of characteristics.

Criteria can be used in many ways in the server. Some of these include:

- When determining which client connection policy should be assigned to a connection
- When determining which log messages should be recorded in an access log
- When determining which types of requests should be allowed over an insecure connection if the `reject-insecure-requests` global configuration property is set to true
- When determining which types of requests should be allowed over an unauthenticated connection if the `reject-unauthenticated-requests` global configuration property is set to true
- When determining which types of bind requests should be forwarded to another server when pass-through authentication is enabled
- When determining which operations should automatically use assured replication.
- When determining which entries should be automatically updated to use entryUUID as the naming attribute
- When determining which types of delete operations should be processed as soft deletes.
- When determining which types of add and modify operations should result in "account created" and "account updated" account status notifications
- When determining the types of operations for which a custom plugin should be invoked

Connection criteria

You can use connection criteria to match client connections based on what the server knows about the connection.

Simple connection criteria

Simple connection criteria is used to match client connections based on a broad set of properties.

These properties include the following.

Property	Description
`included-client-address`	An optional set of address masks (in the form used by the connection handler's `allowed-client` property) that can be used to identify connections that can match this criteria based on the client address.
`excluded-client-address`	An optional set of address masks (in the form used by the connection handler's `allowed-client` property) that can be used to identify connections that do not match this criteria based on the client address.
`included-connection-handler`	An optional set of connection handlers whose connections are allowed to accept connections that can match this criteria.
`excluded-connection-handler`	An optional set of connection handlers whose connections do not match this criteria.
`included-protocol`	An optional set of the communication protocols for connections that might match this criteria.
`excluded-protocol`	An optional set of the communication protocols for connections that do not match this criteria.
`communication-security-level`	The communication security level that can be used by connections that might match this criteria. Possible values include: **`secure-only`** Indicates that this criteria only matches clients that are communicating with the server over a secure connection. **`insecure-only`** Indicates that this criteria only matches clients that are communicating with the server over an insecure connection. **`any`** Indicates that this criteria can match both secure and insecure connections. This is the default value that is used for this property.

Property	Description
`use-auth-type`	The types of authentication that can be used by connections that might match this criteria. By default, this property includes all of the following allowed values: **`none`** Indicates that this criteria can match connections that are not authenticated. ⓘ **Note:** If this value is included, then any authentication-related properties defined in the criteria are ignored for unauthenticated connections. **`simple`** Indicates that this criteria can match connections that are authenticated using LDAP simple authentication. **`sasl`** Indicates that this criteria can match connections that are authenticated using SASL authentication. The `included-user-sasl-mechanism` and `excluded-user-sasl-mechanism` properties can be used to further refine which SASL mechanisms can be used by matching connections. **`internal`** Indicates that this criteria can match connections that were authenticated through an internal mechanism (for example, internal connections created by Server SDK extensions).

Property	Description
authentication-security-level	The types of authentication security that can be used by connections that might match this criteria. This property is ignored for unauthenticated connections. Possible values include: **secure-only** Indicates that this criteria can only match connections in which the authentication was performed in a secure manner. This can include authentication that was processed over a secure connection or in which the authentication can be processed securely even over an insecure connection. **insecure-only** Indicates that this criteria can only match connections in which the authentication was performed in an insecure manner. **any** Indicates that this criteria can match connections in which the authentication was performed in either a secure or insecure manner.
included-user-sasl-mechanism	An optional set of the SASL mechanisms used by clients that can match this criteria. This property is ignored for unauthenticated connections, or connections that did not authenticate with SASL.
excluded-user-sasl-mechanism	An optional set of the SASL mechanisms used by clients that do not match this criteria. This property is ignored for unauthenticated connections, or connections that did not authenticate with SASL.
included-user-base-dn	An optional set of the authenticated user entry base DNs for connections that might match this criteria. This property is ignored for unauthenticated connections.
excluded-user-base-dn	An optional set of the authenticated user entry base DNs for connections that do not match this criteria. This property is ignored for unauthenticated connections.
all-included-user-group-dn	An optional set of the group DNs in which the authenticated user must be a member for connections that might match this criteria. If multiple group DNs are specified, then the authenticated user must be a member of all of those groups. This property will be ignored for unauthenticated connections.

Property	Description
`any-included-user-group-dn`	An optional set of the group DNs in which the authenticated user must be a member for connections that might match this criteria. If multiple group DNs are specified, then the authenticated user must be a member of at least one of those groups. This property is ignored for unauthenticated connections.
`not-all-included-user-group-dn`	An optional set of the group DNs in which the authenticated user should not be a member for connections that might match this criteria. If multiple group DNs are specified, then the authenticated user can optionally be a member of one or more of those groups as long as they are not a member of all of them. This property is ignored for unauthenticated connections.
`none-included-user-group-dn`	An optional set of the group DNs in which the authenticated user must not be a member for connections that might match this criteria. If multiple group DNs are specified, then the authenticated user must not be a member of any of those groups. This property is ignored for unauthenticated connections.
`all-included-user-filter`	An optional set of filters that must match the authenticated user entry for connections that might match this criteria. If multiple filters are specified, then the user entry must match all of them. This property is ignored for unauthenticated connections.
`any-included-user-filter`	An optional set of filters that must match the authenticated user entry for connections that might match this criteria. If multiple filters are specified, then the user entry must match at least one of them. This property is ignored for unauthenticated connections.
`not-all-included-user-filter`	An optional set of filters that should not match the authenticated user entry for connections that might match this criteria. If multiple filters are specified, then the user entry can optionally match one or more of them as long as it does not match all of them. This property is ignored for unauthenticated connections.
`none-included-user-filter`	An optional set of filters that must not match the authenticated user entry for connections that might match this criteria. If multiple filters are specified, then the user entry must not match any of them. This property is ignored for unauthenticated connections.

Property	Description
`all-included-user-privilege`	An optional set of privileges that authenticated users should have for connections that might match this criteria. If multiple privileges are specified, then the user must have all of them. This property is ignored for unauthenticated connections.
`any-included-user-privilege`	An optional set of privileges that authenticated users should have for connections that might match this criteria. If multiple privileges are specified, then the user must have at least one of them. This property is ignored for unauthenticated connections.
`not-all-included-user-privilege`	An optional set of privileges that authenticated users should not have for connections that might match this criteria. If multiple privileges are specified, then the user can optionally have one or more of them as long as it does not have all of them. This property is ignored for unauthenticated connections.
`none-included-user-privilege`	An optional set of privileges that authenticated users should not have for connections that might match this criteria. If multiple privileges are specified, then the user must not have any of them. This property ise ignored for unauthenticated connections.

The default settings for the simple connection criteria match any connection. If you set values for multiple properties, then it essentially behaves as a logical AND, and the criteria only match connections that match all of those properties.

ⓘ **Note:**

A common pitfall encountered with the simple connection criteria is that if the `use-auth-type` property includes none as one of the values, then any properties that pertain to authenticated users are ignored for unauthenticated clients.

A client connection is initially unauthenticated when it is first established and previously authenticated connections might become unauthenticated again if they perform an anonymous bind or if a bind attempt fails.

Aggregate connection criteria

Aggregate connection criteria can be used to create a single connection criteria that combines multiple other connection criteria objects.

It offers the following properties.

Property	Description
`all-included-connection-criteria`	An optional set of the connection criteria objects that must all match the client connection for this aggregate criteria to match.

Property	Description
`any-included-connection-criteria`	An optional set of the connection criteria objects in which at least one of those criteria objects must match the connection for this aggregate criteria to match.
`not-all-included-connection-criteria`	An optional set of the connection criteria objects in which at least one of those criteria objects must not match the connection for this aggregate criteria to match.
`none-included-connection-criteria`	An optional set of the connection criteria objects in which none of those objects can match the connection for this aggregate criteria to match.

Third-party connection criteria
You can use the UnboundID Server SDK to create your own custom connection criteria implementations.

Request Criteria

Use request criteria to match operations based on the content of the request or on the connection on which it was received.

Simple request criteria
Use simple request criteria to match requests based on a broad set of properties.

These properties include the following.

Property	Description
`operation-type`	The set of operation types for requests that might match this criteria. By default, all of the following operation types are included: ▪ `abandon` ▪ `add` ▪ `bind` ▪ `compare` ▪ `delete` ▪ `extended` ▪ `modify` ▪ `modify-dn` ▪ `search` ▪ `unbind`

Property	Description
operation-origin	Specifies the origin for operations that might match this criteria. By default, all of the following operation types are included: **external-request** Indicates that the criteria can match requests from external clients. **internal-operation** Indicates that the criteria can match internal operations, such as those created by Server SDK extensions. **replicated-operation** Indicates that the criteria can match operations received from replication.
connection-criteria	An optional reference to a connection criteria object that must match the connection associated with requests that might match this criteria.
all-included-request-control	An optional set of the OIDs of controls that can be included in requests that might match this criteria. If multiple OIDs are specified, then the request must include all of those controls.
any-included-request-control	An optional set of the OIDs of controls that can be included in requests that might match this criteria. If multiple OIDs are specified, then the request must include at least one of those controls.
not-all-included-request-control	An optional set of the OIDs of controls that should not be included in requests that might match this criteria. If multiple OIDs are specified, then the request can optionally include one or more of those controls as long as it does not have all of them.
none-included-request-control	An optional set of the OIDs of controls that should not be included in requests that might match this criteria. If multiple OIDs are specified, then the request must not include any of them.
included-target-entry-dn	An optional set of base distinguished names (DNs) for entries targeted by requests that match this criteria.
excluded-target-entry-dn	An optional set of base DNs for entries targeted by requests that will not match this criteria.
all-included-target-entry-filter	An optional set of filters that should match the target entry for requests that match this criteria. If multiple filters are specified, then the target entry must match all of them.

Property	Description
`any-included-target-entry-filter`	An optional set of filters that should match the target entry for requests that match this criteria. If multiple filters are specified, then the target entry must match at least one of them.
`not-all-included-target-entry-filter`	An optional set of filters that should not match the target entry for requests that match this criteria. If multiple filters are specified, then the target entry may optionally match one or more of them as long as it does not match all of them.
`none-included-target-entry-filter`	An optional set of filters that should not match the target entry for requests that match this criteria. If multiple filters are specified, then the target entry must not match any of them.
`all-included-target-entry-group-dn`	An optional set of the DNs of groups in which the target entry should be a member for requests that match this criteria. If multiple group DNs are specified, then the target entry must be a member of all of them.
`any-included-target-entry-group-dn`	An optional set of the DNs of groups in which the target entry should be a member for requests that match this criteria. If multiple group DNs are specified, then the target entry must be a member of at least one of them.
`not-all-included-target-entry-group-dn`	An optional set of the DNs of groups in which the target entry should not be a member for requests that match this criteria. If multiple group DNs are specified, then the target entry can optionally be a member of one or more of them as long as it is not a member of all of them.
`none-included-target-entry-group-dn`	An optional set of the DNs of groups in which the target entry should not be a member for requests that match this criteria. If multiple group DNs are specified, then the target entry must not be a member of any of them.

Property	Description
`target-bind-type`	The authentication types for bind requests that can match this criteria. This property is ignored for non-bind requests. By default, this includes both of the following values: **`simple`** Indicates that this criteria can match simple bind requests. **`sasl`** Indicates that this criteria can match SASL bind requests. The `included-target-sasl-mechanism` and `excluded-target-sasl-mechanism` properties can be used to further restrict it to specific mechanisms.
`included-target-sasl-mechanism`	An optional set of the names of SASL mechanisms for SASL bind requests that might match this criteria. This is ignored for all requests other than SASL bind requests.
`excluded-target-sasl-mechanism`	An optional set of the names of SASL mechanisms for SASL bind requests that do not match this criteria. This is ignored for all requests other than SASL bind requests.
`included-target-attribute`	An optional set of the target attributes for requests that might match this criteria. For add requests, the entry to add must include at least one of the target attributes. For compare requests, one of the target attributes must be used in the assertion. For modify requests, at least one modification must include one of the target attributes. For modify DN requests, at least one of the target attributes must be included in the new RDN. For search requests, at least one of the target attributes must be used in the filter. This property is ignored for other types of requests.
`excluded-target-attribute`	An optional set of the target attributes for requests that do not match this criteria.
`included-extended-operation-oid`	An optional set of the OIDs for extended requests that might match this criteria. This is ignored for all requests other than extended requests.
`excluded-extended-operation-oid`	An optional set of the OIDs for extended requests that might not match this criteria. This is ignored for all requests other than extended requests.

Property	Description
`included-search-scope`	The allowed scope values for search requests that might match this criteria. This is ignored for all requests other than search. By default, this includes all of the following values: ▪ `base-object` ▪ `single-level` ▪ `whole-subtree` ▪ `subordinate-subtree`
`using-administrative-session-worker-thread`	Indicates whether to match requests based on their use of a worker thread from the administrative thread pool. The value can be one of the following: **`true`** Indicates that this criteria might only match requests that are processed on an administrative worker thread. **`false`** Indicates that this criteria might only match requests that are not processed on an administrative worker thread. **`any`** Indicates that this criteria may match any type of request, regardless of whether they are processed using an administrative worker thread.
`included-application-name`	An optional set of application names for requests that might match this criteria. The application name for a request can either be specified by either the operation purpose request control or the intermediate client request control. Requests without an application name do not match.
`excluded-application-name`	An optional set of application names for requests that do not match this criteria. Requests with an application name might match this criteria.

The default settings for the simple request criteria match any request. If you set values for multiple properties, then it essentially behaves as a logical `AND`, and the criteria only matches requests that match all of those properties.

ⓘ **Note:**

Properties that are based on the target entry for the request are ignored for abandon and unbind requests, since they do not target a specific entry. They are also ignored for SASL bind and extended requests because the process for determining the target entry, if there is one, depends on decoding that is specific to the type of SASL mechanism or extended request. For search requests, the target entry DN is the search base DN.

Root DSE request criteria
Use the root DSE request criteria type to match requests that target the server root DSE.

The root DSE request criteria type includes the following configuration property:

`operation-type`

Specifies the types of operations that matches the criteria. By default, only the compare and base-object-search types are included. Available values include:

- `compare` — Indicates that the criteria matches compare requests that use the null DN.
- `base-object-search` — Indicates that the criteria matches search requests that use a null base DN with a baseObject scope.
- `single-level-search` — Indicates that the criteria matches search requests that use a null base DN with a singleLevel scope.
- `whole-subtree-search` — Indicates that the criteria matches search requests that use a null base DN with a wholeSubtree scope.
- `subordinate-subtree-search` — Indicates that the criteria matches search requests that use a null base DN with a subordinateSubtree scope.

Aggregate request criteria
Use aggregate request criteria to create a single request criteria that combines multiple other request criteria objects.

Aggregate request criteria offers the following properties.

Property	Description
`all-included-request-criteria`	An optional set of the request criteria objects that must all match the request for this aggregate criteria to match.
`any-included-request-criteria`	An optional set of the request criteria objects in which at least one of those criteria objects must match the request for this aggregate criteria to match.
`not-all-included-request-criteria`	An optional set of the request criteria objects in which at least one of those criteria objects must not match the request for this aggregate criteria to match.
`none-included-request-criteria`	An optional set of the request criteria objects in which none of those objects may match the request for this aggregate criteria to match.

Third-party request criteria
Use the UnboundID Server SDK to create your own custom request criteria implementations.

Result criteria

Use result criteria to match operations based on the content of the result or the associated request or client connection.

Simple result criteria
Use simple result criteria to match results based on a broad set of properties.

These properties include the following.

Property	Description
`request-criteria`	An optional reference to a request criteria object that must match the operation for this criteria to match.
`result-code-criteria`	Specifies the criteria that the operation's result code must match for this criteria to match. Allowed values include: **`all-result-codes`** Any result code can match. This is the default behavior. **`non-failure-result-codes`** Only operations with non-failure result codes (success, compareFalse, compareTrue, referral, saslBindInProgress, and noOperation) can match this criteria. **`failure-result-codes`** Only operations with failure result codes (all result codes other than the non-failure result codes) can match this criteria. **`selected-result-codes`** Only operations with one of the result codes listed in the `result-code-value` property can match this criteria.
`result-code-value`	The set of result codes that an operation might have to match this criteria. This is only used if the `result-code-criteria` property has a value of `selected-result-codes`. Values can include any result code that the PingDirectory Server can use.

Property	Description
processing-time-criteria	Specifies the criteria that the operation's processing time must satisfy for this criteria to match. Allowed values include: **less-than-or-equal-to** Indicates that only operations with a processing time that is less than or equal to the value of the processing-time-value property can match this criteria. **greater-than-or-equal-to** Indicates that only operations with a processing time that is greater than or equal to the value of the processing-time-value property can match this criteria. **any** Indicates that operations with any processing time can match this criteria. This is the default value for this property.
processing-time-value	The duration to use in conjunction with the processing-time-criteria property. This property is ignored if the processing-time-criteria property has a value of any.
queue-time-criteria	Specifies the criteria that the operation's queue time (the time that it spent waiting in the work queue before it was picked up by a worker thread) must satisfy for this criteria to match. Allowed values include: **less-than-or-equal-to** Indicates that only operations with a processing time that is less than or equal to the value of the processing-time-value property can match this criteria. **greater-than-or-equal-to** Indicates that only operations with a processing time that is greater than or equal to the value of the processing-time-value property can match this criteria. **any** Indicates that operations with any processing time can match this criteria. This is the default value for this property.
queue-time-value	The duration to use in conjunction with the queue-time-criteria property. This property is ignored if the queue-time-criteria property has a value of any.

Property	Description
`referral-returned`	Indicates whether this criteria should match operations that included one or more referral URLs. Allowed values include: **`required`** Indicates that this criteria only matches operations that include one or more referral URLs. **`prohibited`** Indicates that this criteria does not match operations that include any referral URLs. **`optional`** Indicates that this criteria can match operations regardless of whether they contain referral URLs. This is the default value for this property.
`all-included-response-control`	An optional set of the OIDs of controls that might be included in responses that can match this criteria. If multiple OIDs are specified, then the operation must include all of those response controls.
`any-included-response-control`	An optional set of the OIDs of controls that might be included in responses that can match this criteria. If multiple OIDs are specified, then the operation must include at least one of those response controls.
`not-all-included-response-control`	An optional set of the OIDs of controls that should not be included in responses that may match this criteria. If multiple OIDs are specified, then the operation can optionally include one or more of those controls as long as it does not include all of them.
`none-included-response-control`	An optional set of the OIDs of controls that should not be included in responses that can match this criteria. If multiple OIDs are specified, then the operation must not include any of those response controls.

Property	Description
`used-alternate-authzid`	Indicates whether this criteria should match operations that used an authorization identity that is different from the authentication identity (for example, as a result of the proxied authorization request control). Allowed values include: **`required`** Indicates that this criteria only matches operations that include one or more referral URLs. **`prohibited`** Indicates that this criteria does not match operations that include any referral URLs. **`optional`** Indicates that this criteria can match operations regardless of whether they contain referral URLs. This is the default value for this property.
`used-any-privilege`	Indicates whether this criteria should match operations in which the requester used one or more privileges. **`required`** Indicates that this criteria only matches operations in which the requester used one or more privileges. **`prohibited`** Indicates that this criteria only matches operations in which the requester did not use any privileges. **`optional`** Indicates that this criteria can match operations regardless of whether the requester used any privileges.
`used-privilege`	An optional set of the privileges that the requester might have used for this criteria to match the operation. If multiple privileges are specified, then the requester must have used at least one of those privileges.

Property	Description
`missing-any-privilege`	Indicates whether this criteria should match operations in which the requester was missing one or more required privileges. **`required`** Indicates that this criteria only matches operations in which the requester was missing one or more privileges. **`prohibited`** Indicates that this criteria only matches operations in which the requester was not missing any privileges. **`optional`** Indicates that this criteria can match operations regardless of whether the requester was missing any required privileges.
`missing-privilege`	An optional set of the privileges that were required for the associated operation that the requester must have been missing for this criteria to match. If multiple privileges are specified, then the requester must have been missing at least one of those privileges.
`retired-password-used-for-bind`	Indicates whether this criteria should match bind operations based on whether the client authenticated with a retired password. This property is ignored for all operations other than bind. Allowed values include: **`retired-password-used`** Indicates that this criteria only matches bind operations in which the user authenticated with a retired password. **`retired-password-not-used`** Indicates that this criteria only matches bind operations in which the user did not authenticate with a retired password. **`any`** Indicates that this criteria can match bind operations regardless of whether the user authenticated with a retired password. This is the default value for the property.

Property	Description
search-entry-returned-criteria	Specifies the criteria for the number of entries returned in response to a search operation that operations matching this criteria should use. This property is ignored for all operations other than search. Allowed values include: **equal-to** Indicates that this criteria only matches search operations in which the number of entries returned matches the value of the search-entry-returned-count property. **not-equal-to** Indicates that this criteria only matches search operations in which the number of entries returned does not match the value of the search-entry-returned-count property. **greater-than-or-equal-to** Indicates that this criteria only matches search operations in which the number of entries returned is greater than or equal to the value of the search-entry-returned-count property. **less-than-or-equal-to** Indicates that this criteria only matches search operations in which the number of entries returned is less than or equal to the value of the search-entry-returned-count property. **any** Indicates that this criteria can match search operations regardless of the number of entries that were returned. This is the default value for the property.
included-authz-user-base-dn	An optional set of base DNs below which the operation's authorization identity might exist for the criteria to match.
excluded-authz-user-base-dn	An optional set of base DNs below which the operation's authorization identity must not exist for the criteria to match.
all-included-authz-user-group-dn	An optional set of the DNs of groups in which the operation's authorization identity must be a member for this criteria to match. If multiple group DNs are specified, then the user must be a member of all of those groups.

Property	Description
`any-included-authz-user-group-dn`	An optional set of the DNs of groups in which the operation's authorization identity must be a member for this criteria to match. If multiple group DNs are specified, then the user must be a member of at least one of those groups.
`not-all-included-authz-user-group-dn`	An optional set of the DNs of groups in which the operation's authorization identity should not be a member for this criteria to match. If multiple group DNs are specified, then the user can optionally be a member of one or more of those groups as long as it is not a member of all of them.
`none-included-authz-user-group-dn`	An optional set of the DNs of groups in which the operation's authorization identity should not be a member for this criteria to match. If multiple group DNs are specified, then the user must not be a member of any of them.

The default settings for the simple result criteria matches any operation. If you set values for multiple properties, then it essentially behaves as a logical `AND`, and the criteria only matches operations that match all of those properties.

Replication assurance result criteria
Use replication assurance result criteria to match operations based on their use of assured replication, whether based on a request control or criteria, and whether the constraints were satisfied.

Property	Description
`local-assurance-level`	The set of local replication assurance levels that can be in use for the criteria to match. By default, this property has all three of the following possible values: **`none`** Indicates that the criteria should match if the operation does not expect any degree of local assurance. **`received-any-server`** Indicates that the criteria should match if the operation expects that the change will be received by at least one other local server before the response is returned to the client. **`processed-all-servers`** Indicates that the criteria should match if the operation expects that the change will be processed by all other available local servers before the response is returned to the client.

Property	Description
`remote-assurance-level`	The set of remote replication assurance levels that might be in use for the criteria to match. By default, this property has all four of the following possible values: **`none`** Indicates that the criteria should match if the operation does not expect any degree of remote assurance. **`received-any-remote-location`** Indicates that the criteria should match if the operation expects that the change will be received by at least one server in at least one remote location before the response is returned to the client. **`received-all-remote-locations`** Indicates that the criteria should match if the operation expects that the change will be received by at least one server in all of the remote locations before the response is returned to the client. **`processed-all-remote-servers`** Indicates that the criteria should match if the operation expects that the change will be processed by all available servers in all remote locations before the response is returned to the client.
`assurance-timeout-criteria`	Indicates the criteria that should be used to match the assurance timeout. Allowed values include: **less-than-or-equal-to** Indicates that the criteria can match if the assurance timeout is less than or equal to the value contained in the `assurance-timeout-value` property. **`greater-than-or-equal-to`** Indicates that the criteria can match if the assurance timeout is greater than or equal to the value contained in the assurance-timeout-value property. **`any`** Indicates that the assurance timeout is not considered when deciding whether the criteria matches the operation. This is the default value for the property.

Property	Description
`assurance-timeout-value`	The duration value used in conjunction with the `assurance-timeout-criteria`. This is ignored if the assurance-timeout-criteria value is any.
`response-delayed-by-assurance`	Indicates whether the criteria should match an operation based on whether the operation response was delayed by replication assurance processing. Allowed values include: **`true`** Indicates that the criteria can match if the operation response was delayed by assurance processing. **`false`** Indicates that the criteria does not match if the operation response was delayed by assurance processing. **`any`** Indicates that the criteria does not depend on whether the response was delayed by replication assurance processing. This is the default value for the property.
`assurance-behavior-altered-by-control`	Indicates whether the criteria should match an operation based on whether it included an assured replication request control that used different assurance criteria than the server would have otherwise used for the operation. **`true`** Indicates that the criteria can match if the operation included an assured replication request control with different criteria than would otherwise be used. **`false`** Indicates that the criteria can match if the operation did not include an assured replication request control or if it included a control with the same behavior that the server would have used without the control. **`any`** Indicates that the criteria does not depend on whether the assurance behavior was altered by a control. This is the default value for the property.

Property	Description
`assurance-satisfied`	Indicates whether the criteria should match the operation based on whether the assurance constraints were satisfied. **`both-satisfied`** Indicates that the criteria can match if both the local and remote assurance requirements were satisfied. **`either-satisfied`** Indicates that the criteria can match if one or both of the local and remote assurance requirements were satisfied. **`at-least-local-satisfied`** Indicates that the criteria can match if the local assurance requirements were satisfied, regardless of whether the remote requirements were satisfied. **`at-least-remote-satisfied`** Indicates that the criteria can match if the remote assurance requirements were satisfied, regardless of whether the local requirements were satisfied. **`only-local-satisfied`** Indicates that the criteria can match if the local assurance requirements were satisfied, but the remote requirements were not satisfied. **`only-remote-satisfied`** Indicates that the criteria can match if the remote assurance requirements were satisfied, but the local requirements were not satisfied. **`either-not-satisfied`** Indicates that the criteria can match if one or both of the local or remote requirements were not satisfied. **`at-least-local-not-satisfied`** Indicates that the criteria can match if the local assurance requirements were not satisfied, regardless of whether the remote requirements were satisfied. **`at-least-remote-not-satisfied`** Indicates that the criteria can match if the remote assurance requirements were not satisfied, regardless of whether the local requirements were satisfied. **`neither-satisfied`** Indicates that the criteria can match if neither the local nor the remote criteria were satisfied. **`any`** Indicates that the criteria does not depend

Aggregate result criteria
Use aggregate result criteria to create a single result criteria that combines multiple other result criteria objects.

Aggregate result criteria offers the following properties.

Property	Description
`all-included-result-criteria`	An optional set of the result criteria objects that must all match the result for this aggregate criteria to match.
`any-included-result-criteria`	An optional set of the result criteria objects in which at least one of those criteria objects must match the result for this aggregate criteria to match.
`not-all-included-result-criteria`	An optional set of the result criteria objects in which at least one of those criteria objects must not match the result for this aggregate criteria to match.
`none-included-result-criteria`	An optional set of the result criteria objects in which none of those objects might match the result for this aggregate criteria to match.

Third-party result critera
Use the UnboundID Server SDK to create your own custom result criteria implementations.

Search entry criteria

Use search entry criteria when matching search result entries, which are entries that match search criteria and are intended to be returned to the client.

They contain the DN and set of attributes that you typically find in an entry, and they can also optionally include a set of controls.

Simple search entry criteria
Simple search entry criteria objects provide support for matching search result entries based on the following properties.

Property	Description
`request-criteria`	An optional reference to request criteria that is expected to match the search request for this entry criteria to match.
`all-included-entry-control`	An optional set of OIDs of controls that must be included with the search result entry to match this criteria. If multiple OIDs are specified, then the entry must have all of those controls.
`any-included-entry-control`	An optional set of OIDs of controls that must be included with the search result entry to match this criteria. If multiple OIDs are specified, then the entry must have at least one of those controls.

Property	Description
`not-all-included-entry-control`	An optional set of OIDs of controls that should not be included with the search result entry to match this criteria. If multiple OIDs are specified, then the entry can optionally have one or more of those controls as long as it does not have all of them.
`none-included-entry-control`	An optional set of OIDs of controls that should not be included with the search result entry to match this criteria. If multiple OIDs are specified, then the entry must not have any of those controls.
`included-entry-base-dn`	An optional set of base DNs for entries that might match this criteria.
`excluded-entry-base-dn`	An optional set of base DNs for entries that do not match this criteria.
`all-included-entry-filter`	An optional set of OIDs of search filters that must match the entry for this criteria to match. If multiple filters are specified, then the entry must match all of them.
`any-included-entry-filter`	An optional set of OIDs of search filters that must match the entry for this criteria to match. If multiple filters are specified, then the entry must match at least one of them.
`not-all-included-entry-filter`	An optional set of OIDs of search filters that should not match the entry for this criteria to match. If multiple filters are specified, then the entry can optionally match one or more of them as long as it does not match all of them.
`none-included-entry-filter`	An optional set of OIDs of search filters that should not match the entry for this criteria to match. If multiple filters are specified, then the entry must not match any of them.
`all-included-entry-group-dn`	An optional set of the DNs of groups in which the entry must be a member for this criteria to match. If multiple groups are specified, then the entry must be a member of all of them.
`any-included-entry-group-dn`	An optional set of the DNs of groups in which the entry must be a member for this criteria to match. If multiple groups are specified, then the entry must be a member of at least one of them.
`not-all-included-entry-group-dn`	An optional set of the DNs of groups in which the entry should not be a member for this criteria to match. If multiple groups are specified, then the entry can optionally be a member of one or more of them as long as it is not a member of all of them.

Property	Description
`none-included-entry-group-dn`	An optional set of the DNs of groups in which the entry should not be a member for this criteria to match. If multiple groups are specified, then the entry must not be a member of any of them.

Aggregate search entry criteria
Use aggregate search entry criteria to create a single search entry criteria that combines multiple other search entry criteria objects.

It offers the following properties.

Property	Description
`all-included-search-entry-criteria`	An optional set of the search entry criteria objects that must all match the entry for this aggregate criteria to match.
`any-included-search-entry-criteria`	An optional set of the search entry criteria objects in which at least one of those criteria objects must match the entry for this aggregate criteria to match.
`not-all-included-search-entry-criteria`	An optional set of the search entry criteria objects in which at least one of those criteria objects must not match the entry for this aggregate criteria to match.
`none-included-search-entry-criteria`	An optional set of the search entry criteria objects in which none of those objects can match the entry for this aggregate criteria to match.

Third-party search entry criteria
Use the UnboundID Server SDK to create your own custom search entry criteria implementations.

Search reference criteria

Use search reference criteria when matching search result references, which provide information about other servers or different portions of the DIT in which the client can continue the search. They contain a set of referral URLs and an optional set of controls.

Simple search reference criteria
Simple search reference criteria objects provide support for matching search result references based on the following properties.

Property	Description
`request-criteria`	An optional reference to request criteria that is expected to match the search request for this reference criteria to match.
`all-included-reference-control`	An optional set of OIDs of controls that must be included with the search result reference to match this criteria. If multiple OIDs are specified, then the reference must have all of those controls.

Property	Description
`any-included-reference-control`	An optional set of OIDs of controls that must be included with the search result reference to match this criteria. If multiple OIDs are specified, then the reference must have at least one of those controls.
`not-all-included-reference-control`	An optional set of OIDs of controls that should not be included with the search result reference to match this criteria. If multiple OIDs are specified, then the reference may optionally have one or more of those controls as long as it does not have all of them.
`none-included-reference-control`	An optional set of OIDs of controls that should not be included with the search result reference to match this criteria. If multiple OIDs are specified, then the reference must not have any of those controls.

Aggregate search reference criteria
Use aggregate search reference criteria to create a single search reference criteria that combines multiple other search reference criteria objects.

It offers the following properties.

Property	Description
`all-included-search-reference-criteria`	An optional set of the search reference criteria objects that must all match the reference for this aggregate criteria to match.
`any-included-search-reference-criteria`	An optional set of the search reference criteria objects in which at least one of those criteria objects must match the reference for this aggregate criteria to match.
`not-all-included-search-reference-criteria`	An optional set of the search reference criteria objects in which at least one of those criteria objects must not match the reference for this aggregate criteria to match.
`none-included-search-reference-criteria`	An optional set of the search reference criteria objects in which none of those objects can match the reference for this aggregate criteria to match.

Third-party search reference criteria
Use the UnboundID Server SDK to create your own custom search reference criteria implementations.

Authentication

Authentication is the process that a client uses to identify themselves to the server.

This includes three basic components:

- Identify the account that is trying to authenticate. This is often provided in the form of a DN or username, but it can come in other forms like a Kerberos principal or information in a certificate or OAuth bearer token.
- Provide proof of that identity. This can include a password, optionally combined with a second factor like a one-time password, a certificate, or interaction with a trusted system like a Kerberos KDC or an OAuth introspection endpoint.
- Optionally specify an alternate authorization identity. Usually, when a client authenticates, subsequent operations on that connection are processed using the authority of the authenticated user. However, in some cases, it might be possible for the client to request that subsequent operations be processed under the authority of a different user.

For LDAP clients, the PingDirectory Server supports both simple authentication and several SASL mechanisms. For HTTP clients, the server supports basic authentication and OAuth 2.0.

This section covers the set of authentication mechanisms that the PingDirectory Server supports. It also provides information about its extensive support for password policy functionality, some of which also applies to non-password-based authentication.

LDAP simple authentication

The most common type of authentication for LDAP clients is the simple bind.

When using simple authentication, clients identify themselves by providing the distinguished name (DN) of the entry for their account, and they prove their identity with a password.

The password is provided to the server in the clear, so it is especially vital to protect the communication using TLS. Because simple binds are a single-factor authentication mechanism relying only on the password as proof of identity, it is important to ensure that the password is strong and stored in a secure manner.

It is also possible to perform anonymous authentication with an LDAP simple bind. In this case, the bind DN and password should both be empty. Although the original LDAPv3 specification, *RFC 2251*, indicated that it was possible to perform an anonymous simple bind with just an empty password, allowing for a non-empty DN, this resulted in many security problems in LDAP-based applications that didn't verify that the password was non-empty. The revised LDAPv3 specification, *RFC 4511*, discourages allowing a non-empty DN with an empty password. By default, the PingDirectory Server rejects bind attempts with an empty password but non-empty DN.

SASL authentication

Simple Authentication and Security Layer (SASL) is a standard authentication framework described in *RFC 4422*.

It isn't any one type of authentication, but it is an extensible framework that allows for just about any type of authentication. The type of authentication is identified with a mechanism name, and the credentials can be encoded in a manner that is specific to that mechanism.

Some SASL mechanisms provide support for specifying an alternate authorization identity that should be used for processing subsequent operations on the connection. Some mechanisms also provide support for adding integrity (digital signatures) or confidentiality (encryption) to any communication that occurs over the connection after the authentication has completed.

Standard SASL mechanisms

There are a number of Simple Authentication and Security Layer (SASL) mechanisms defined in various RFCs and IETF drafts. This section details the standard mechanisms that the PingDirectory Server supports.

PLAIN

The PLAIN mechanism is a password-based authentication mechanism that is described in *RFC 4616* and is similar to LDAP simple authentication. However, it provides two advantages that simple authentication doesn't:

- LDAP simple authentication only allows you to identify the client by providing the distinguished name (DN) of the entry for their account. The PLAIN SASL mechanism allows you to identify the client with either a username or a DN.
- The PLAIN mechanism optionally allows you to specify an alternate authorization identity that should be used to authorize subsequent requests on the connection.

If you want to authenticate by a username, then you should prefix that username with "`u:`" such as `u:jdoe` for a username of `jdoe`. The server uses an identity mapper to identify the entry that corresponds to that username. If you want to authenticate with a DN, then you should prefix the DN with `dn:`, such as `dn:uid=jdoe, ou=People, dc=example, dc=com`.

As when using LDAP simple authentication, the PLAIN mechanism provides the password to the server in the clear. It should therefore only be used over a secure connection.

CRAM-MD5 and DIGEST-MD5

CRAM-MD5 and DIGEST-MD5 are password-based authentication mechanisms that allow the client to identify themselves with either a username prefixed by `u:` or a DN prefixed by `dn:`.

The primary benefit that these mechanisms offer over the PLAIN mechanism is that they do not send the password to the server in the clear. Instead, they send an MD5 digest that combines the username, password, and some randomly generated data. The DIGEST-MD5 mechanism optionally also allows you to specify a realm and an alternate authorization identity.

While this might initially seem like an improvement over the PLAIN mechanism, there are two key reasons that these mechanisms are discouraged:

- They rely on the MD5 message digest algorithm, which has long been considered weak and should no longer be relied upon to secure communication.
- Even though the bind request doesn't include the clear-text password, it requires that both the client and the server have access to the clear-text password. This means that the password must be stored in the server in a reversible form, which greatly increases the risk of exposure in the event of a data breach.

Because of these security concerns, both of these mechanisms have officially declared historic through *draft-ietf-sasl-crammd5-to-historic* for CRAM-MD5 and *RFC 6331* for DIGEST-MD5) and should no longer be used. Although the PingDirectory Server supports them, we strongly discourage their use.

EXTERNAL

The EXTERNAL SASL mechanism described in *RFC 4422* authenticates a client based on information that is available for that client outside of information transmitted over LDAP. To do this, PIngDirectory Server uses a certificate chain that the client presents to the server during TLS negotiation. The server uses a certificate mapper to identify the user that is authenticating. It can optionally also require the public portion of the certificate to be present in the user's entry.

Because the EXTERNAL mechanism in the PingDirectory Server can only occur over a TLS-encrypted connection, it is a secure mechanism. Although it only uses a single authentication factor, as long as the server is configured with an appropriate trust manager to properly validate the client certificate chain, it is a much stronger form of authentication than a password.

GSSAPI

The GSSAPI SASL mechanism described in *RFC 4752* allows clients to authenticate with a Kerberos KDC. The credentials, typically either a password or a key read from a keytab file, are validated by the KDC and are not exposed to the PingDirectory Server. The PingDirectory server does need to have an account for the user (and an identity mapper is used to map the Kerberos principal to an entry in the server), but that account does not need to include a password if you are only going to authenticate with GSSAPI.

The Kerberos authentication process protects the credentials in transit so that they are not exposed to network observers. Further, it provides an option to protect communication on the connection that occurs after authentication with either integrity protection that digitally signs the communication to protect it from being altered in transit but does not prevent it from being observed, or confidentiality protection, that encrypts the communication so that it cannot be observed or altered. Alternatively, GSSAPI authentication can occur over a TLS-encrypted connection to protect all communication on that connection.

See the `config/sample-dsconfig-scripts/enable-gssapi-authentication.dsconfig` batch file for more information about configuring GSSAPI authentication in the PingDirectory Server.

OAUTHBEARER

The OAUTHBEARER SASL mechanism described in *RFC 7628* allows a client to authenticate using an OAuth 2.0 bearer token. The access token can be obtained from an OAuth authorization server, which can authenticate the client through a variety of means, such as a username and password, a two-factor mechanism that combines a static password with a one-time password, or a FIDO hardware token. The PingDirectory Server uses an access token validator to ensure that the token is authentic and to map it to an entry in the server.

The token can also be associated with one or more OAuth scopes. These are associated with the user's authentication state and can be used in conjunction with the `oauthscope` access control bind rule to grant or deny access control rights to the user.

ANONYMOUS

As its name implies, the ANONYMOUS mechanism described in *RFC 4505* only performs anonymous authentication. That is, it does not identify any account, and it does not provide any proof of identity.

The bind request can include an optional trace string in the SASL credentials, which can provide additional information about the context for the bind (for example, the application that is associated with the connection), but this should be taken with a grain of salt because without any proof of identity, there is nothing to prevent the client from lying about what it includes in the trace string.

The PingDirectory Server supports the ANONYMOUS SASL mechanism although it is disabled by default because it provides very little benefit over anonymous simple authentication.

Proprietary SASL mechanisms
In addition to the standard SASL mechanisms listed earlier, the PingDirectory server provides support for a number of proprietary mechanisms.

UNBOUNDID-CERTIFICATE-PLUS-PASSWORD

The UNBOUNDID-CERTIFICATE-PLUS-PASSWORD mechanism is similar to the EXTERNAL mechanism in that it allows a client to authenticate with a certificate presented during TLS negotiation. However, it adds a second authentication factor in the form of a static password. This makes it a two-factor mechanism that adds an additional layer of protection in the event that the user's client certificate is compromised or if an attacker is somehow able to get a trusted certificate that maps to the user entry.

UNBOUNDID-DELIVERED-OTP

The UNBOUNDID-DELIVERED-OTP mechanism allows a user to authenticate with the combination of a static password and a one-time password (OTP) that is generated by the server and delivered to the

user through some out-of-band mechanism. The PingDirectory Server provides out-of-the-box support for delivering the one-time password through an SMS, through the *Twilio* service, text message, or in an email message, but you can create custom OTP delivery mechanisms using the UnboundID Server SDK.

When using the UNBOUNDID-DELIVERED-OTP mechanism, the client first uses the deliver one-time password extended request, which sends a username and static password to the server as preliminary proof of identity. The server then generates a one-time password and deliver it to the user through an appropriate means. Finally, the client includes the authentication ID, the delivered one-time password, and an optional authorization identity in an UNBOUNDID-DELIVERED-OTP bind request to complete the authentication process.

The one-time password that the server generates is only valid for a limited period of time, five minutes by default. After they are generated, these one-time passwords are stored in the user's entry. The "Encrypt TOTP Secrets and Delivered Tokens" plugin can be used to ensure that these values are encrypted when stored in the server.

See the `config/sample-dsconfig-batch-files/enable-delivered-otp-authentication.dsconfig` batch file for more information about configuring the UNBOUNDID-DELIVERED-OTP SASL mechanism.

UNBOUNDID-EXTERNALLY-PROCESSED-AUTHENTICATION

The UNBOUNDID-EXTERNALLY-PROCESSED-AUTHENTICATION mechanism is unusual in that it doesn't actually change the authentication state of the underlying client connection. The connection must already be authenticated as a user that has either the `permit-externally-processed-authentication` privilege or the `proxied-auth` privilege, and it remains authenticated as that user regardless of the outcome of the UNBOUNDID-EXTERNALLY-PROCESSED-AUTHENTICATION bind attempt.

The real benefit of the UNBOUNDID-EXTERNALLY-PROCESSED-AUTHENTICATION mechanism is that it allows an application to authenticate a user through some external means, but still verify that authentication in the PingDirectory Server. It can ensure that the user's account is in a usable state, such as not disabled or locked, and it can also update the user's password policy state to do things like update the user's login history and increment the set of failed attempts and possibly lock the account in the event that the external authentication attempt failed.

UNBOUNDID-TOTP

The UNBOUNDID-TOTP mechanism is a two-factor authentication scheme that allows a user to identify themselves with a username or DN and verify that identity with both a static password and a time-based one-time password generated using the TOTP algorithm described in *RFC 6238*. The bind request can optionally include an alternate authorization identity.

Time-based one-time passwords are generated using a variety of free software, including phone apps like Google Authenticator and Authy as well as desktop software. The software used to generate TOTP passwords depends only on the current time and a secret key that is shared between the client and the server. There is no need to communicate with any other system, so it's an especially good choice for administrative accounts because it allows them to use strong authentication that work even if the server is unable to communicate with other systems. TOTP authentication does require that the client and server both agree on what time it is, but it does allow for some leeway. By default, the server allows the TOTP generator's clock to be at least a minute head of or a minute behind the server's clock.

The client and server need to be using the same shared secret to generate the time-based one-time passwords. The "generate TOTP shared secret" extended operation can be used to generate a suitable secret value, store it in the user's entry, and return it to the client to allow them to configure it for use with their authenticator app. The "revoke TOTP shared secret" extended operation can be used to revoke a secret if it is no longer needed or if you are concerned that it might have been compromised.

In some cases, the client might want to verify a time-based one-time password without using the UNBOUNDID-TOTP SALS mechanism. For example, the client can implement its own kind of "step up"

authentication in which a simple password might be sufficient for some operations but additional proof of identity might be required when performing other operations. To help with this, the server offers a “validate TOTP password” extended operation that allows the client to provide the user’s DN and TOTP password to verify for that user.

> ⓘ **Note:**
>
> This extended operation does not actually change the authentication state for the underlying connection.

The shared secret that the server uses to generate TOTP shared secrets needs to be stored in a reversible form so that the server can use it to verify the values that the client provides. The server can also store the last TOTP password that the client used to prevent it from being used more than once. The “Encrypt TOTP Secrets and Delivered Tokens” plugin can be used to ensure that these values are encrypted when stored in the server.

UNBOUNDID-YUBIKEY-OTP

The UNBOUNDID-YUBIKEY-OTP mechanism is a two-factor authentication scheme that allows a user to identify themselves with a username or DN and verify their identity with the combination of a static password and a one-time password generated by a YubiKey device offered by *Yubico*.

The one-time password that is generated can be sent to the public validation servers that Yubico maintains or you can run your own local authentication servers. Although running your own servers requires additional effort, it eliminates a dependency on a third-party service and also allows you to use YubiKey devices that have been updated to use a different private key than was initially assigned to them.

The “register YubiKey OTP device” extended operation can be used to register a device with the server and associate it with a user account. The server also offers a corresponding “deregister Yubikey OTP device” extended operation that can remove information about the device from the user’s entry.

See the `config/sample-dsconfig-batch-files/enable-yubikey-otp-authentication.dsconfig` batch file for more information on this SASL mechanism.

Third-Party SASL Mechanisms
Use the UnboundID Server SDK to develop support for custom SASL mechanisms that the server should support.

HTTP client authentication

Clients that are communicating with the server over HTTP can authenticate in one of two ways:

HTTP basic authentication

The client provides a simple username and password. An identity mapper is used to identify the entry, and the password is used to prove that identity.

An OAuth bearer token

The client provides the server with an OAuth 2.0 bearer token and the server uses an access token validator to verify that the token is authentic and map it to a user’s entry.

The set of authentication methods used depends on the endpoint with which the client is communicating.

Pass-through authentication

The PingDirectory Server also provides support for pass-through authentication in which the client sends a simple bind request to the local server, and the server can forward the request to another server to actually verify the credentials.

The server allows the authentication attempt to be passed through to two different types of servers:

Another LDAP directory server

This can be useful when migrating to the PingDirectory Server from another type of directory server, especially if that server does not provide any way to export the authentication credentials from that server. This can be enabled through the pass-through authentication plugin.

The cloud-based *PingOne* service

This can be useful when migrating between an on-premise PingDirectory Server and the PingOne service. This can be enabled through the PingOne pass-through authentication plugin.

Both of these pass-through authentication mechanisms offer a similar set of options, including:

- The bind attempt can be tried locally first, or you can have it always pass through the authentication attempt to the backend service.
- The plugin can optionally update the password in the user's entry if the local authentication attempt fails, but the credentials are accepted by the service to which the request is passed through.
- You can use criteria to indicate which bind requests should be passed through to the backend service.

> ⓘ **Note:**
>
> Only one pass-through authentication plugin instance can be enabled in the server at any one time.

Identity mapping

The PingDirectory Server provides an identity mapper framework that allows it to identify the user entry that corresponds to a provided identifier such as a username or a Kerberos principal.

Out-of-the-box support is provided for two types of identity mappers:

Exact Match

The server performs an internal search to find entries in which the provided identifier exactly matches the value of one of a specified set of attributes in the user's entry. The default instance of the exact match identity mapper is configured to match any user entry whose `uid` or `mail` attribute contains a value that matches the provided identifier. For example, if the provided identifier is "`jdoe`", then the identity mapper would perform an internal search with a filter of "`(|(uid=jdoe)(mail=jdoe))`".

Regular Expression

The server uses a regular expression to transform the provided identifier in some way, and then looks for an entry that contains the resulting value in one of a specified set of attributes. The default instance of the regular expression identity mapper is configured to strip off an at sign and anything after it in the provided username, and then to search for any entries that have the resulting string as a value for the `uid` attribute. For example, if the provided identifier is "`jdoe@EXAMPLE.COM`", then the mapper would perform an internal search with a filter of "`(uid=jdoe)`".

It is also possible to use the UnboundID Server SDK to create custom identity mapper implementations if those provided by the server are not sufficient.

The identity mapper must be able to identify exactly one entry that corresponds to the given identifier. If it cannot find any appropriate entries, or if it finds multiple matching entries, then the identity mapping attempt fails.

Certificate mapping

The PingDirectory Server uses a component called a certificate mapper to identify the user entry that corresponds to a given certificate, such as in the course of processing a bind using the EXTERNAL or UNBOUNDID-CERTIFICATE-PLUS-PASSWORD SASL mechanism.

The types of certificate mappers that it offers by default include:

Subject Equals DN

This certificate mapper expects the subject DN of the certificate to match the distinguished name (DN) of the corresponding user entry.

Subject Attribute to User Attribute

This certificate mapper extracts the values of a specified set of attributes from the certificate subject and search for an entry containing those values in a corresponding set of attributes. The default instance of this certificate mapper tries to map the CN attribute from the certificate's subject to the `cn` attribute in the user's entry, or the `E` attribute in the certificate's subject to the mail attribute in the user's entry.

Subject DN to User Attribute

This certificate mapper expects the user's entry to contain a specified attribute whose value matches the subject DN of the presented certificate.

Fingerprint

This certificate mapper expects the user's entry to contain a specified attribute whose value matches the SHA-256, SHA-1, or MD5 fingerprint of the presented certificate.

You can also use the UnboundID Server SDK to create custom certificate mapper implementations.

Using alternate authorization identities

Under certain conditions, a client that is authenticated as one user can request that the server process operations under the authority of a different user.

The most common ways to accomplish this include:

- The client can use the proxied authorization request control. The server supports two variants of this control:
  - A version that is described in *draft-weltman-ldapv3-proxy-04*, in which the target user is identified by a distinguished name (DN).
  - A version that is described in *RFC 4370*, in which the target user is identified by an authorization ID that can include either a username (prefixed by `u:`) or a DN (prefixed by `dn:`).
- The client can include an intermediate client request control that contains a value in the clientIdentity field. This value should be an authorization ID that starts with either `u:` or `dn:`.
- Some SASL mechanisms allow the user to request an alternate authorization identity that will automatically be used for all subsequent operations requested on that connection. This includes the DIGEST-MD5, GSSAPI, OAUTHBEARER, PLAIN, UNBOUNDID-DELIVERED-OTP, UNBOUNDID-TOTP, and UNBOUNDID-YUBIKEY-OTP mechanisms.

This is a very powerful mechanism, and it allows applications to efficiently support multiple concurrent users over a common set of pooled connections. These pooled connections can be authenticated with an account that is specific to that application, and then each time that application needs to perform an operation on behalf of one of its end users, it can include the proxied authorization or intermediate client control in the request.

However, there are also obvious security concerns associated with this ability. If any user could impersonate any other user, then access controls would be irrelevant. PingDirectory Server offers several forms of protection to ensure that alternate authorization identities can be used safely. These include:

- Only clients who have the `proxied-auth` privilege are permitted to use an alternate authorization identity. This privilege is not granted to any user by default (even root users).
- The use of the proxied authorization and intermediate client controls can be restricted by access control.
- Operational attributes in user entries can be used to restrict the use of alternate authorization identities. See *Restricting access through operational attributes in user entries* on page 1717 for more information about these operational attributes.

- Only accounts with a usable password policy state can be used as alternate authorization identities. For example, if an account is administratively disabled, locked because of too many failed authentication attempts, has an expired password, or has any other state that would prevent them from authenticating to the server, then the server does not allow operations to be processed under the authority of that user.

The retain identity request control

Bind operations alter the authentication state of the underlying client connection. If the bind attempt succeeds, then subsequent operations are processed under the authority of the authenticated user or possibly an alternate authorization identity for some SASL mechanisms.

If the bind attempt fails, then the connection reverts to an unauthenticated state. This behavior is dictated by LDAP specifications, and it is the appropriate behavior in most circumstances.

However, it might not be ideal for some applications, and especially those that support multiple concurrent users and use Directory Server to authenticate those users. In such cases, connection pooling allows existing connections to be reused for multiple operations (and even operations that need to be processed under the authority of different users), which is much more efficient and resource-friendly than establishing a new connection for each request or maintaining a separate connection for each authenticated user. In such cases, performing a bind operation to verify a user's identity might affect the application's ability to reuse the connection for other purposes.

The most common way to address this in applications is to maintain two pools of connections. One that is used for only processing bind operations to verify user credentials, and a second that is used to process all other types of operations. However, PingDirectory Server offers an alternative that only requires the application to maintain a single pool: the *retain identity request control*. If this control is included in a bind request, then the server performs all of the usual processing for the bind operation, including identifying the user, verifying their credentials, and applying any necessary updates to their password policy state, but it does not affect the authentication state of the underlying connection. Regardless of whether the bind attempt succeeds or fails, the connection remains authenticated as the same account with the same authorization identity that it had before the bind was attempted.

Delaying responses to failed bind attempts

Configure PingDirectory Server to delay the response to bind requests as a way of rate-limiting online password guessing attacks.

This can be done in two different ways:

- The LDAP connection handler offers a `failed-bind-response-delay` configuration property. If this is set to a nonzero duration, then the server automatically delays the response to any failed bind attempt by the specified length of time. The server does not delay the response to successful bind attempts.
- The password policy offers a `failure-lockout-action` configuration property that can be used to indicate what action should be taken after too many failed authentication attempts, and one possible action is delaying the bind response. For more information, see This will be covered in more detail in the *Failure lockout* section in the discussion on password policies.

The option to delay bind responses in the connection handler was available before the corresponding option in the password policy. However, the latter option is the recommended approach because it also delays the response to the first successful bind following several failed attempts, which makes it more difficult for an attacker to use the delay to identify a failed attempt and to abort early without waiting for the full failure duration. You can also configure the password policy approach to work for non-LDAP clients.

Password policies

Because password policies govern many aspects of the server's behavior, especially with regard to authentication and password management, it is critical to configure them appropriately.

Assigning password policies to users

Each user is associated with a password policy that governs their activity in the server, and different users can have different password policies.

Whenever the server processes an operation that attempts to authenticate a user, change their password, or interact with their password policy state in some way, it needs to select an appropriate password policy to use for that processing.

A user can be associated with a password policy through the `ds-pwp-password-policy-dn` operational attribute. If this attribute exists in the user's entry and refers to a valid password policy, then the user is subject to that password policy. If that attribute exists but refers to a nonexistent policy, then that user is unable to authenticate or be used as an alternate authorization identity. If a user's entry does not include the `ds-pwp-password-policy-dn` operational attribute, then that user is subject to the server's default password policy, which is specified by the default-password-policy property in the global configuration.

The `ds-pwp-password-policy-dn` operational attribute can be either real or virtual. You can explicitly set a value for the attribute in a user's entry, but it is also possible to have the server generate a value for that attribute based on some criteria using the virtual attribute subsystem. For example, you could use a virtual attribute to automatically assign the same password policy to all members of a specified group or to all users in a specified portion of the DIT.

> ⓘ **Note:**
>
> A user should not be conditionally subjected to different password policies under different circumstances. While it is technically possible to use virtual attributes that assign different values to the same attribute under different conditions, this capability should not be used for the `ds-pwp-password-policy-dn` attribute.

For example, you should not attempt to detect which application has issued a request and select a password policy based on that application. The server only maintains one set of password policy state for each user, and attempting to access the same user under different password policies might have unexpected adverse effects and can introduce security risks.

Maintaining password policies in user data

While password policies are typically defined in the server configuration, it is also possible to define them in user data.

This is particularly useful when the PingDirectory Server is used to back a multi-tenant application, in which information about many different organizations is maintained in the same Directory Server instance, typically with a separate branch for each organization. You can configure password policies on a per-organization basis.

Each password policy must be defined in an entry containing the `ds-cfg-password-policy` structural object class. The definition of this object class can be found by querying the server schema over LDAP, by retrieving the "`cn=schema`" entry, or by looking in the `config/schema/02-config.ldif` schema definition file. The names of the LDAP attribute types which should correspond to names of the password policy configuration properties that are available in dsconfig or the administration console with a "`ds-cfg-`" prefix.

For example, the following entry represents a minimal password policy that might be defined in user data.

```
dn: cn=Org X Password Policy,ou=Org X,ou=tenants,dc=example,dc=com
```

```
objectClass: top
objectClass: ds-cfg-password-policy
cn: Org X Password Policy
ds-cfg-password-attribute: userPassword
ds-cfg-default-password-storage-scheme: cn=Salted SHA-256,cn=Password
 Storage
  Schemes,cn=config
```

Assign users to password policies contained in the user data in the same way that you assign them to policies in the configuration. Include the `ds-pwp-password-policy-dn` operational attribute in their entry as either a real or a virtual attribute.

> ⓘ **Note:**
>
> While password policies can reside in user data, any other configuration elements that they reference, including password storage schemes, password validators, password generators, account status notification handlers, and failure lockout actions, must reside within the configuration.

For improved performance, the PingDirectory Server maintains a cache of password policy entries that are defined in the user data. This cache holds up to 500 policies by default, but you can tune that through the `maximum-user-data-password-policies-to-cache` property in the global configuration.

Password storage schemes

The PingDirectory Server does not store passwords in the clear.

Whenever a password is set through an add operation, a modify operation, or a password modify extended operation, and when clear-text passwords are included in entries imported from LDIF, the server uses a component called a password storage scheme to encode that password. Whenever a client attempts a password-based bind, the password storage scheme determines whether the password included in the bind request matches one stored in the user's entry.

The server allows users to authenticate with passwords encoded using any scheme that the server supports. However, when storing new passwords, it uses the `default-password-storage-scheme` property in the password policy configuration to determine which scheme to use when encoding that password. Although it is technically possible to have multiple default schemes enabled simultaneously within the same password policy, only enable one scheme at a time unless you need to synchronize encoded passwords to multiple different systems that require different encodings.

Supported password storage schemes

PingDirectory Server supports many password storage schemes. The most notable of these schemes are described here.

Salted variants of the SHA-2 digest

The SHA-2 algorithms described in *RFC 6234* are a suite of cryptographic one-way digest algorithms that can be used to generate 256-bit, 384-bit, and 512-bit hashes of arbitrary amounts of data. One-way digests means that there is no known way to examine the resulting hash and determine the clear-text value used to generate it, other than simply trying every possible combination of clear-text values until you find one that matches. It is also computationally infeasible to generate collisions, that is, to find two different clear-text values that result in the same hash.

The SHA-2 digest algorithms are considered secure although there are two issues that can make them suboptimal on their own. The first is that providing the same clear-text input to the digest algorithm consistently yields the same hash. This makes it vulnerable to precomputed dictionary attacks in which attackers might use a dictionary that maps hashes to the clear-text values used to generate them and also makes it possible to determine whether two different users have the same password. This weakness can be addressed by combining the passwords with some amount of randomly generated data (called a salt)

before computing the digest and then making the salt available in addition to the digest. PingDirectory Server only offers support for salted versions of the SHA-2 password storage schemes.

The second issue is that SHA-2 digests are fast to compute on their own. It is not unreasonable for an attacker to have access to hardware capable of generating billions of SHA-2 digests per second. This means that it can be faster to try large numbers of possibilities in an attempt to guess the password used to generate a digest through brute force. While strong passwords can still be highly resistant to this form of attack, weak passwords (for example, those using a smaller number or range of characters, and especially those based on dictionary words) can be quickly broken. Some other password storage schemes attempt to address this by requiring multiple rounds of digest computation, requiring parallel processing, or requiring large amounts of memory, but even in those cases, the best defense is still a strong password and ideally one that is combined with a second factor like a one-time password.

PingDirectoryServer uses the salted 256-bit SHA-2 digest as the default password storage scheme for regular users, that is, for users other than root users and topology administrators.

Salted and unsalted variants of the SHA-1 digest

The SHA-1 algorithm described in *RFC 3174* is a cryptographic one-way digest algorithm that was in widespread use for many years as a leading mechanism for encoding passwords, and some directory servers in widespread use can still use it as the basis for their default password storage schemes. However, researchers have been able to devise attacks against the SHA-1 digest that make it possible to generate collisions under some circumstances (albeit with considerable computational effort). As such, it is no longer recommended for use in encoding new passwords.

PingDirectory Server supports both salted and unsalted variants of the SHA-1 digest. Support for the salted variant is enabled by default, but you should only use it as a deprecated scheme for the purpose of migrating passwords to a more secure alternative, like one of the SHA-2 digests or one of the PBKDF2, Argon2, bcrypt, or scrypt key derivation functions. Support for the unsalted SHA-1 digest is disabled by default and should only be enabled if needed for compatibility with legacy systems.

The PBKDF2 key derivation function

The PBKDF2 (password-based key derivation function, version 2, as described in *RFC 2898*) is a standard algorithm for encoding passwords and generating cryptographic keys from passwords. The PBKDF2 algorithm uses a salt to make it resistant to precomputed dictionary attacks, and it also uses multiple rounds of computation to increase the amount of processing required to compute the encoded representation for the password and slow down the rate at which attackers can make guesses in brute-force attacks.

> ⓘ **Note:**
>
> While the PBKDF2 function is a well-defined standard, the specification only covers the algorithm used to derive a key using the password, salt, and number of rounds as inputs.

There is no standard encoding that combines the resulting digest with the salt and number of rounds used to generate that digest. This means that while multiple products can support using the PBKDF2 algorithm to encode passwords, the way they represent the encoded passwords might be different. PingDirectory Server uses an encoding that attempts to be as compact as possible, helping to minimize the on-disk and in-memory footprint for the data, and as a result it is very likely to be different from the encoding used by any other product. The encoding that we use is constructed as follows:

1. The first byte represents the encoding version and the message digest algorithm. Possible values include:
   - 0x00 — The SHA-1 digest (PBDKDF2WithHmacSHA1)
   - 0x01 — The 256-bit SHA-2 digest (PBDKDF2WithHmacSHA256)
   - 0x02 — The 384-bit SHA-2 digest (PBDKDF2WithHmacSHA384)
   - 0x03 — The 512-bit SHA-2 digest (PBDKDF2WithHmacSHA512)
2. The second byte represents the number of bytes used to hold the salt. The storage scheme supports salts from 8–127 bytes (64–1016 bits). By default, the server generates 16-byte (128-bit) salts.
3. The byte used to indicate the salt length is followed immediately by the bytes that comprise the salt.
4. The next two or four bytes holds the iteration count used for the algorithm. If the iteration count is less than or equal to 32,767, then it is encoded in two bytes. If the iteration count is greater than or equal to 32,768, then it is encoded in four bytes, and the most significant bit of the first byte is set to 1.
5. The remainder of the encoded password is the output of the PBKDF2 algorithm generated from the clear-text password and salt using the indicated algorithm and iteration count.

The bytes resulting from the above steps are base64-encoded, and the whole string is preceded by a prefix of "`{PBKDF2}`".

> ⓘ **Note:**
>
> Even though the encoding that the PingDirectory Server uses might differ from that of other servers, as long as those other servers use the same standard algorithm to derive the key, it should still be possible to convert between their encoded representations and the one that the PingDirectory Server uses. It should therefore be possible to migrate PBKDF2-encoded passwords between systems.

The PingDirectory Server uses the PBKDF2 password storage scheme by default for root users and topology administrators.

See the `config/sample-dsconfig-batch-files/use-pbkdf2-password-storage-scheme.dsconfig` batch file for more information on configuring and using the PBKDF2 password storage scheme.

The Argon2, bcrypt, and scrypt key derivation functions

The Argon2, bcrypt, and scrypt key derivation functions are strong, highly regarded functions for encoding passwords. They are all intentionally expensive as a means of making them more resistant to password guessing attacks. The bcrypt algorithm uses a cost factor to increase the amount of processing required. The script and Argon2 algorithms also employ memory access and parallel processing as additional techniques for increasing the cost of generating large numbers of passwords.

Unlike the PBKDF2, the Argon2, bcrypt, and scrypt algorithms do have predefined encodings that combine all of the necessary metadata like the salt, cost factors, and other settings used to compute the resulting key. As such, the format that PingDirectory Server uses for passwords encoded with these schemes should be compatible with the format used by other servers that support them.

> ⓘ **Note:**
>
> Support for the Argon2, bcrypt, and scrypt password storage schemes depends on the third-party Bouncy Castle cryptographic provider library. To simplify compliance with U.S. export regulations, PingDirectory Server does not include this library by default, but it can be obtained for free from the Bouncy Castle website (*https://www.bouncycastle.org/*) and placed in the server's lib directory to make it available for use.

See the `use-argon2-password-storage-scheme.dsconfig`, `use-argon2-password-storage-scheme.dsconfig`, and `use-scrypt-password-storage-scheme.dsconfig` files in the `config/sample-dsconfig-batch-files` directory for more information about configuring and using these password storage schemes.

The crypt password storage scheme

PingDirectory Server also provides support for a suite of password encodings that match those often used by Linux and UNIX-based systems. This includes the legacy UNIX crypt algorithm which uses the DES encryption algorithm and an extremely small 12-bit salt, as well as stronger, multi-round variants that are based on the MD5 and SHA-2 digests. It can support authentication with any of these variants, but it only uses one of them when encoding new passwords.

Only the 256-bit and 512-bit SHA-2 variants are considered secure, and they are the only ones that should be used for encoding new passwords. The 256-bit SHA-2 variant is used by default. However, because of the potential to support the weak DES-based and MD5-based algorithms, we discourage the use of the crypt password storage scheme unless absolutely required for compatibility.

The AES encryption algorithm

PingDirectory Server also supports storing passwords in a reversibly encrypted form using the AES cipher algorithm. Because this storage scheme uses reversible encryption, there is a much greater risk that passwords encoded with this scheme could be exposed in the event of a data breach. As such, use of this password storage scheme is highly discouraged unless you need to use an authentication mechanism that requires access to the clear-text password, such as the CRAM-MD5 or DIGEST-MD5 SASL mechanisms.

Custom password storage schemes

Use the UnboundID Server SDK to implement support for custom password storage schemes using whatever encoding you want. The Server SDK provides two types of extensions for this purpose:

- The `PasswordStorageScheme` class provides support for encoding and validating passwords using only the clear-text password and information in the configuration for that storage scheme.
- The `EnhancedPasswordStorageScheme` class also provides access to other information in the user's entry, as well as a potential set of updates that are being applied to that entry. This option might be required if the password encoding depends on other information stored in the user's entry, such as if the salt or other encoding metadata is stored separately from the encoded password.

A custom password storage scheme might be required if you need to migrate data from another system into a PingDirectory Server instance, and the migrated data includes passwords encoded in a format that PingDirectory Server does not support.

Fast algorithms versus expensive algorithms

Some password encoding algorithms, like the SHA-1 and SHA-2 schemes, can be cheap to compute, while others like PBKDF2, Argon2, bcrypt, and scrypt can be expensive to compute.

The more expensive a password storage scheme is, the more effort that an attacker must expend when trying to crack passwords encoded with that scheme.

However, the higher resistance to password cracking attacks does not come for free. If it is more expensive for attackers to generate encoded password representations, it is also more expensive for the server to generate them in response to legitimate requests. Because the server has to generate these encoded representations whenever it is processing a bind request to authenticate a user, or when processing an add, modify, or password modify request used to set a new password, using these storage schemes can significantly affect server performance.

You can tune each of these expensive schemes in some way to adjust the performance impact that it will have. If you plan to use one of them, determine what level of authentication and password change performance (in terms of number of operations per second) that you will need, and the minimum cost factor that you are willing to accept for the password storage scheme. Then, benchmark the system to make sure that your environment can meet those demands or to determine how much additional hardware you might require to do so.

This performance impact can be even more significant when importing data that contains clear-text passwords. Ideally, if you're migrating data from another system, that system also has been storing passwords in a relatively secure manner, and you will therefore not have access to their clear-text

representations. However, if you do have access to the clear-text passwords, or if you're importing sample data for testing purposes, then you can find the import proceeding at a crawl because of the expensive password processing. If you're just using sample data for testing purposes, then pre-encode the passwords so that the import can proceed as quickly as possible. If you're using real-world data, then you might want to use a faster password storage scheme (like one of the SHA-2 schemes) for the import, and then configure that scheme as deprecated (as described in the next section) so that the passwords are gradually migrated to the preferred encoding.

Alternatively, you might decide that the potential benefit you get from using an expensive password storage scheme as increased resistance to password cracking attacks is not worth the added processing cost that it incurs for legitimate processing. This can be a completely valid conclusion to reach in some environments, especially if you take steps to help ensure that users are required to choose strong passwords and continue to monitor the strength of those passwords over time. Using a more expensive encoding might require an attacker to expend thousands of times more resources to crack passwords than if they had used a fast encoding, but this additional cost is largely irrelevant if users have weak passwords that can be quickly guessed by attackers.

On the other hand, using a very fast algorithm might not incur that much risk for sufficiently strong passwords. For example, if a ten-character password is comprised of a random combination of uppercase and lowercase letters, numeric digits, and ASCII symbols, a set of 95 characters, then there are 9510 = 59,873,693,923,837,890,625 possible combinations of those characters. Even if an attacker can make one hundred billion guesses per second, it would still take nearly nineteen years for them to try every option. Realistically, users probably aren't going to have ten-character completely random passwords, but you can still do things like encourage password managers, encourage passphrases, discourage password reuse, and prohibit known-weak passwords. You can also use two-factor authentication so that gaining access to a user's account requires more than just cracking their password.

Ideally, you might find that the constraints of your budget and performance requirements do not prohibit the use of expensive encoding and that you can enjoy the best of both worlds: strong passwords encoded in a secure manner and protected by an additional authentication factor

Deprecated password storage schemes
PingDirectory Server provides support for deprecating password storage schemes.

If a storage scheme is configured as deprecated, which can be done using the `deprecated-password-storage-scheme` property in the password policy configuration, any user who authenticates using a password encoded with that scheme using a mechanism that provides the server with access to the clear-text representation of that password automatically has their password re-encoded using the default scheme. This provides an excellent way to transparently migrate user passwords from weaker encodings to stronger ones without requiring users to change their passwords.

See the `config/sample-dsconfig-batch-files/deprecate-weak-password-storage-schemes.dsconfig` batch file for more information about deprecating password storage schemes.

Pre-encoded passwords
Whenever possible, passwords to be stored in PingDirectory Server should be provided in the clear.

This is likely not an option when migrating data from an existing system into PingDirectory Server, as passwords from those systems are likely already encoded in a non-reversible form. However, passwords included in add, modify, and password modify operations should be provided in the clear. If users are allowed to provide passwords in pre-encoded form, then they might be able to exempt themselves from some of the server's password quality constraints, including:

- The server cannot validate a pre-encoded password to ensure that it meets password quality requirements. Allowing pre-encoded passwords can therefore allow users to choose weak passwords.
- The server cannot ensure that a pre-encoded password is not just an alternative representation of the same password that they are currently using (for example, one that just uses a different salt), or a password that is already in their history. Allowing pre-encoded passwords can therefore allow users to circumvent restrictions around password expiration and password reuse.

- The server cannot ensure that passwords are encoded using the desired scheme. Allowing pre-encoded passwords can permit clients to use encodings that are not as resistant to password cracking attacks, and can also make it harder to migrate to migrate to stronger schemes in the future.

By default, PingDirectory Server does not permit clients to provide pre-encoded passwords. While this restriction can be lifted using the `allow-pre-encoded-passwords` in the password policy configuration, we discourage this for the reasons listed above. In the event that there are legitimate use cases in which some applications might need to set pre-encoded passwords, such as when synchronizing passwords from another system into the PingDirectory Server, there are a couple of better alternatives:

- You can update the account used by any such applications to grant them the `bypass-pw-policy` privilege. This privilege exempts the user from certain password policy restrictions when setting new passwords for other users, but not when changing the password for their own account. This includes:
  - They are permitted to set pre-encoded passwords.
  - They are permitted to set passwords that would otherwise fail password validation.
  - They are permitted to set passwords that are already in the user's password history.
- The client can use the password update behavior request control to customize the behavior that the server exhibits for the associated operation.

Password validators

Password validators ensure that user passwords are of sufficient quality.

When processing an add operation, a modify operation, or a password modify extended operation that attempts to set a new password, the server can reject that operation if the password is deemed too weak by one or more of the password validators. It is also possible to invoke password validators when users authenticate in a manner that provides the server with access to their clear-text password as an ongoing means of ensuring password quality.

Supported password validators

The PingDirectory server provides support for a wide variety of commonly used password validators.

The attribute value password validator

The attribute value password validator can be used to help ensure that users are not allowed to choose passwords that match other information in their entry. For example, a user might want to make their password the same as their username or email address. If an attacker gains access to information in a user's entry, then they can use that information to customize attacks gained at cracking that user's password.

The attribute value password validator offers the following configuration options:

`match-attribute`

The names of the other attributes to check when validating the password. If this is not provided, then all attributes are used by default.

`test-password-substring-of-attribute-value`

Indicates whether to reject passwords that are a substring of the value of another attribute in the user's entry. By default, only exact matches are rejected.

`test-attribute-value-substring-of-password`

Indicates whether to reject passwords that contain the value of another attribute in the user's entry as a substring. By default, only exact matches are rejected.

`minimum-attribute-value-length-for-substring-matches`

An integer value that specifies the minimum attribute value length that will be checked if `test-attribute-value-substring-of-password` is set to true. This can help prevent inappropriately rejecting passwords that happen to contain short attribute values. For example,

if you have an attribute whose values can be just a single character, you probably don't want to prevent passwords that happen to contain that character. By default, only attributes values that contain at least four characters are checked.

`test-reversed-password`

Indicates whether to check attribute values in reverse order in addition to the order in which the values are stored. For example, if an attribute has value of "jdoe", then this could cause the server to also check for a value of "eodj". This is true by default.

See the `config/sample-dsconfig-batch-files/use-the-attribute-value-password-validator.dsconfig` batch file for more information about using the attribute value password validator.

The character set password validator

The character set password validator can be used to ensure that passwords contain an appropriate combination of characters. For example, you might want to require that a password contain at least one lowercase letter, one uppercase letter, one digit, and one symbol. Or you might want to ensure that a password contains characters from at least three of those four sets.

The character set password validator offers the following configuration properties:

`character-set`

Defines the sets of characters that are validated. Each of these sets should be defined as an integer followed by a colon and the characters contained in that set. The integer defines the minimum number of characters from that set that must be present in passwords. For example, a `character-set` value of "`1:abcdefghijklmnopqrstuvwxyz`" indicates that passwords will be required to have at least one lowercase letter. The integer value can be zero to indicate that characters from that set are permitted to be included in passwords but are not required. By default, the validator is configured with sets containing all lowercase letters, all uppercase letters, all numeric digits, and a range of ASCII symbols, and to require passwords to contain at least one character from each of those sets.

`allow-unclassified-characters`

Indicates whether to allow passwords to contain characters that are not contained in any of the defined character sets. This is true by default.

`minimum-required-character-sets`

Specifies the minimum number of character sets that are required to be included in passwords. This is useful, for example, if you want to require passwords to contain characters from at least three of the four predefined sets of lowercase letters, uppercase letters, digits, and symbols. In this case, all of the character sets should be updated to have a minimum required count (the value before the colon) of zero to indicate that they are all optional. If any character set has a nonzero minimum required count, then at least that many characters from that set are required in passwords, even if they would otherwise have contained characters from enough of the other sets.

ⓘ **Note:**

While this has been and still is a common practice, it is discouraged by modern password guidelines, such as those in *NIST Special Publication 800-63B section 5.1.1.2*. Studies have shown that these types of constraints annoy users without significantly improving password security, or even giving the false impression of security.

See the `config/sample-dsconfig-batch-files/use-the-character-set-password-validator.dsconfig` batch file for more information about using the character set password validator.

The dictionary password validator

The dictionary password validator can be used to prevent passwords that are known to be weak. The server comes predefined with two instances of the dictionary password validator:

- One that contains a dictionary of common words from a variety of languages
- One that contains a dictionary of strings that are known to be the most commonly used passwords

We also recommend that you provide your own dictionary that includes additional words not included in either of the included word lists. In particular, we recommend prohibiting words that are related to your company's name or products or services that you provide.

The dictionary password validator offers the following configuration properties:

dictionary-file

The path to a file containing the list of prohibited passwords. Each line of the file represents a prohibited password.

case-sensitive-validation

Indicates whether differences in capitalization should not be ignored when checking passwords against the dictionary file. By default, the validator uses case-insensitive validation.

test-reversed-password

Indicates whether to test passwords in reversed order when checking them against the dictionary file. For example, to prohibit a password of "terces" if the dictionary contains the word "secret". Reversed passwords are not checked by default.

ignore-leading-non-alphabetic-characters

Indicates whether to ignore any non-alphabetic characters that might appear at the start of the password when checking passwords against the dictionary file. For example, "123secret" could be rejected if the dictionary contains the word "secret". By default, all leading characters are considered significant.

ignore-trailing-non-alphabetic-characters

Indicates whether to ignore any non-alphabetic characters that might appear at the end of the password when checking passwords against the dictionary file. For example "secret123" could be rejected if the dictionary contains the word "secret". By default, all trailing characters are considered significant.

strip-diacritical-marks

Indicates whether to strip diacritical marks, such as accents, cedillas, circumflexes, diaereses, tildes, and umlauts, off of characters before checking passwords against the dictionary. For example, "sèçrét" could be rejected if the dictionary contains the word "secret". By default, diacritical marks are preserved.

alternative-password-character-mapping

Specifies a set of alternative characters that might be substituted in when checking passwords against a dictionary. For example, if a dollar sign is a substitute for the letter s, the number 3 is a substitute for the letter e, and the number 7 is a substitute for the letter t, then "$3cr37" could be rejected if the dictionary contains the word "secret". By default, no character substitutions are defined.

maximum-allowed-percent-of-password

Indicates that the proposed password should be rejected if it makes up at least a given percentage of a dictionary word . For example, if this threshold is set to 70%, then "mysecret" could be rejected if the dictionary contains the word "secret" because the word "secret" makes up 75% of "mysecret". By default, this is set to 100%, so values are ony rejected if they match complete dictionary words

(after performing any other configured processing, like stripping of leading or trailing non-alphabetic characters).

See the `config/sample-dsconfig-batch-files/use-the-dictionary-password-validator.dsconfig` batch file for more information about using the dictionary password validator.

The haystack password validator

The haystack password validator is based on the concept of password haystacks as described at *https://www.grc.com/haystack.htm*. It is based on the premise that a password's resistance to brute-force attacks comes from a combination of two factors:

- The number of characters contained in the password. Longer passwords require more effort to crack than shorter passwords.
- The types of characters contained in the password. Passwords containing a wider range of characters might require more effort to crack than shorter passwords. For example, a password containing a mix of lowercase and uppercase letters might take longer to crack than a password containing just lowercase letters (at least, if an attacker crafts their attack to prioritize passwords that contain characters from only one set over those that contain passwords from multiple sets)

Because this cost is a combination of two factors, an increase in one of them can make up for a deficiency in another. For example, a password containing only lowercase letters can be very strong if it is long enough, while a shorter password might be acceptable if it contains a wider mix of characters.

This is a useful metric to offer because it can reject passwords that are legitimately easy to crack while not flatly rejecting passwords purely based on the types of characters that they contain. Unfortunately, it can be a difficult concept to succinctly convey to users when describing the requirements that their password will have to satisfy, which might ultimately make it undesirable for use.

The haystack password validator offers the following configuration properties:

`assumed-password-guesses-per-second`

The assumed rate at which an attacker can make password guessing attempts. By default, the validator assumes a rate of one hundred billion guesses per second.

`minimum-acceptable-time-to-exhaust-search-space`

The minimum acceptable length of time required to try all possible combinations of characters in the sets that make up the proposed password at the configured maximum rate. By default, the minimum acceptable time is one week.

See the `config/sample-dsconfig-batch-files/use-the-haystack-password-validator.dsconfig` batch file for more information about using the haystack password validator.

The length-based password validator

The length-based password validator can be used to impose restrictions on the allowed number of characters that a password may contain. Available configuration properties include:

`min-password-length`

The minimum number of characters that passwords are allowed to contain. Passwords that contain fewer characters than this limit are rejected.

`max-password-length`

The maximum number of characters that passwords are allowed to contain. Passwords that contain more characters than this limit are rejected. A value of zero indicates that no limit is enforced.

In some cases, it might be useful to impose a maximum length to prevent the server from spending too much processing power on extremely long passwords, but we strongly discourage setting a maximum

length that is too short and unnecessarily limits the strength that a password can have. *NIST Special Publication 800-63B section 5.1.1.2* recommends allowing passwords to be at least 64 characters long.

The PingDirectory server includes two length-based dictionary validator definitions by default: one with a minimum length of six characters and another with a minimum length of twelve characters. The NIST guidelines referenced above recommend requiring that passwords contain at least 8 characters.

See the `config/sample-dsconfig-batch-files/use-the-length-based-password-validator.dsconfig` batch file for more information about using the length-based password validator.

The Pwned Passwords password validator

The Pwned Passwords validator can be used to prevent users from choosing passwords that are known to have been compromised. By default, it checks with the free *Pwned Passwords service*, which includes a database of hundreds of millions of compromised passwords although it can be used with any other service that uses a compatible API.

The Pwned Passwords service uses the k-Anonymity algorithm to allow a client to securely determine whether a password has been compromised without revealing what the proposed password actually is. It does this by first computing an unsalted SHA-1 digest of the proposed password, and then only sending the first five characters of the hexadecimal representation of that digest to the service. The service then identifies all compromised passwords whose hex-encoded SHA-1 digest starts with those five characters and return the remaining 35 characters for all of those digests back to the client. If that returned set includes the last 35 characters in the digest for the proposed password, then it is one that has been known to be compromised.

The Pwned Passwords validator offers the following configuration properties:

`pwned-passwords-base-url`

The base URL to use for the service with which to communicate. By default, this is *https://api.pwnedpasswords.com/range/*.

`invoke-for-add`

Indicates whether the validator should be invoked for passwords set in add requests. Thisis true by default.

`invoke-for-self-change`

Indicates whether the validator should be invoked for new passwords chosen by the end user. This is true by default.

`invoke-for-admin-reset`

Indicates whether the validator should be invoked for new passwords set by an administrator. This is true by default.

`accept-password-on-service-error`

Indicates whether the validator should consider the password acceptable if it cannot communicate with the target service, or if that service returns an error response. By default, the password is considered acceptable although it might still be rejected if it fails to satisfy one or more of the other configured validators.

`key-manager-provider`

A key manager provider that should be used for HTTPS communication with the target service. This can be omitted, which is the case by default, if no key store is required because the client does not need to present its own client certificate chain to the server.

`trust-manager-provider`

A trust manager that should be used for HTTPS communication to determine whether to trust the server's certificate chain. This can be omitted if the server should use the JVM's default trust store.

See the `config/sample-dsconfig-batch-files/use-the-pwned-passwords-validator.dsconfig` batch file for more information about using the Pwned Passwords password validator.

The regular expression password validator

The regular expression password validator may be used to require passwords to match a given regular expression, or to reject passwords that do not match a given regular expression. It offers the following configuration properties:

`match-pattern`

The regular expression pattern to use when evaluating proposed passwords.

`match-behavior`

The behavior to exhibit if the given pattern matches a proposed password. Allowed values include:

- `require-match` — Indicates that the validator should accept passwords that match the provided pattern and reject passwords that do not.
- `reject-match` — Indicates that the validator should reject passwords that match the provided pattern and accept passwords that do not.

The repeated characters password validator

The repeated characters password validator can be used to reject passwords that have the same character more than a specified number of times in a row. This can help prevent passwords that contain stretches of the same character repeated several times, such as “aaaaaaaa”.

The repeated characters password validator offers the following configuration properties:

`max-consecutive-length`

The maximum number of times that the character is allowed to be present consecutively within a proposed password. The default value is 2, which means that it is acceptable for the same character to appear twice in a row, but not three times or more.

`case-sensitive-validation`

Indicates whether to treat uppercase and lowercase versions of the same letter as different. By default, the validator uses case-insensitive validation, which means that it would reject a password containing the string “aAa” if the validator is configured with a max-consecutive-length of 2.

`character-set`

An optional set of characters that should be considered equivalent for the purpose of this password validator. For example, you could use this to define a set of numeric digits, and then reject passwords that contain too many consecutive digits.

Although many organizations may use a requirement that rejects passwords with repeated characters, we do not recommend using this validator unless it is configured with a longer limit than the default (for example, four characters rather than two). A limit that is too small may unnecessarily prevent very strong long randomly generated passwords (for example, one created by a password manager) just because the same character happened to occur a few times in a row. However, the unique characters password validator can be used to achieve the same result in a better way, so its use is recommended over the repeated characters validator.

The similarity-based password validator

The similarity-based password validator can be used to prevent users from choosing new passwords that are too similar to their current password. It uses the Levenshtein distance algorithm to compute the

minimum number of changes (characters added to, removed from, or replaced in) to convert the current password into the new password.

This password validator is only invoked for self-changes. It is not invoked for add operations or administrative password resets. Further, because it requires access to the user's current password, it should only be used if the `password-change-requires-current-password` property is set to true in the password policy configuration.

This password validator offers the following configuration properties:

`min-password-difference`

The minimum number of changes (character insertions, removals, or replacements) required to convert the user's current password into the proposed new password for that new password to be considered acceptable. The default value is 3.

See the `config/sample-dsconfig-batch-files/use-the-similarity-based-password-validator.dsconfig` batch file for more information about using the similarity-based password validator.

The unique characters password validator

The unique characters password validator can be used to reject passwords that contain fewer than a specified minimum number of unique characters. This can help prevent users from choosing passwords that only contain a few different characters, such as "abababab".

This password validator supports the following configuration properties:

`min-unique-characters`

The minimum number of different characters that can appear in an acceptable password. Passwords that contain fewer than this number of unique characters are rejected.

`case-sensitive-validation`

Indicates whether to perform case-sensitive validation, in which uppercase and lowercase versions of the same letter are treated as different characters. By default, the validator uses case-insensitive validation.

See the `config/sample-dsconfig-batch-files/use-the-unique-characters-password-validator.dsconfig` batch file for more information about using the unique characters password validator.

Custom password validators

The UnboundID Server SDK can be used to create custom password validators using whatever logic you want.

Configuring password validators for updates

Password policies contain the following properties for configuring the set of password validators that are invoked for add operations, modify operations, and password modify extended operations.

`password-validator`

The set of password validators that should be invoked. Zero or more validators can be configured.

`skip-validation-for-administrators`

Indicates whether to allow administrators to set passwords that do not satisfy the password validation requirements.

No validators are included in the out-of-the-box configuration for the default password validator. Unless the `--allowWeakRootUserPassword` argument is provided when running setup, or the equivalent

option is chosen when setting up the server in interactive mode, the passwords for root users and topology administrators are subject to the following requirements:

- The password must contain at least 12 characters.
- The password must not be contained in a dictionary of common words in various languages
- The password must not be contained in a dictionary of commonly used passwords

The `skip-validation-for-administrators` property is false by default in both the default password policy and the policy for root users and topology administrators.

Configuration password validators for binds

The PingDirectory Server also provides support for invoking password validators while processing bind operations in which the server has access to the user's clear-text password.

This can be useful in at least a couple of key circumstances:

- If you use a service like Pwned Passwords that is regularly updated with information from new data breaches
- If you maintain one or more dictionaries of banned passwords and update them with new values
- If you enable a new validator or change the configuration of an existing validator

> ⓘ **Note:**
>
> The set of password validators that are configured for add, modify, and password modify operations is not automatically used for bind operations. Rather, you can explicitly configure which validators (if any) should be invoked for binds.

Similarly, password validators do not have to be invoked for every bind. In many cases, it may be better to only periodically invoke validators to reduce the potential impact on performance. This is especially true for cases in which validation may interact with external systems like the Pwned Passwords service.

Password policies provide the following configuration properties related to invoking password validators on bind:

`bind-password-validator`

The set of password validators to invoke for bind operations. Zero or more validators can be configured, and none are configured by default.

`minimum-bind-password-validation-frequency`

The minimum frequency that can be used when invoking validators for users. If this is set to zero seconds, then password validators will be invoked for each bind. If it is set to a value that is greater than zero, then the server keeps track of the last time it invoked bind password validators for that user and only invokes the validators if at least that much time has passed since the last set of validation. The default validation frequency is 30 days.

`bind-password-validation-failure-action`

The action that the server should take if a user's password fails validation during a bind. Allowed values include:

- `force-password-change` — Indicates that the bind attempt should succeed, but the user will be forced to choose a new password before they are allowed to perform any other operations in the server. The bind response includes a password expired response control and a diagnostic message that describes the problems identified with the password. This is the default behavior.
- `reject-bind` — Indicates that the bind attempt should fail and the user's account should be locked. An administrative password reset is required to restore access to the user's account.
- `generate-account-status-notification` — Indicates that the bind attempt should succeed, but that the server should generate an account status notification to make the user or

server administrators aware of the weakness, or potentially to take custom action in response to it.

See the `config/sample-dsconfig-batch-files/enable-password-validation-for-binds.dsconfig` batch file for more information about enabling password validators for binds.

Recommended password validator configuration
To help ensure that users choose strong passwords, configure the following password validators for add, modify, and password modify operations.

- A length-based password validator with a minimum length of ten characters.
- A dictionary password validator configured to use `config/wordlist.txtA`.
- A dictionary password validator configured to use `config/commonly-used-passwords.txt`.
- A dictionary password validator configured to use a custom dictionary with banned words that relate to the company and its products or services, as well as any other banned passwords that are not included in dictionaries shipped with the PingDirectory Server.
- The Pwned Passwords validator that prohibits passwords from known data breaches.
- An attribute value password validator that rejects passwords matching attribute values.
- A similarity-based password validator that rejects new passwords that are too similar to the current password.
- A unique characters password validator that requires passwords to contain at least five unique characters.

We also recommend periodically invoking the following password validators for bind operations:

- The Pwned Passwords validator that prohibits passwords from known data breaches.
- The dictionary validator that uses the custom banned password file if you intend to update it on a regular basis with new banned passwords.

Consider forcing users to change their passwords if validation fails during a bind operation.

Password history

Configure PingDirectory Server to maintain a history of former passwords to prevent them from reusing the password multiple times.

Use the following password policy configuration properties to enable a password history:

`password-history-count`

The maximum number of former passwords to maintain in the history.

`password-history-duration`

The maximum length of time that former passwords should be stored in the history.

If either of these properties is configured with a nonzero value, then the server maintains a password history for users associated with that password policy.

If a password history is to be maintained, then you might want to also impose a limit on how frequently users are allowed to change their password. Without such a limit, some crafty users might attempt to change their passwords several times in quick succession to purge the password they want to keep from the history so they can re-use it. Configure this limit with the following configuration property:

`min-password-age`

The minimum length of time that must pass between self password changes. If a user attempts to change their password multiple times within this duration, then the latter attempts are rejected.

> ⓘ **Note:**

Administrators are able to reset user passwords at any time, regardless of how long it has been since a user has changed their password. It also does not prevent a user from choosing a new password following an administrative reset.

See the `config/sample-dsconfig-batch-files/enable-password-history.dsconfig` batch file for more information about enabling password history.

Password expiration

While it was once a common practice, and still is in some environments, password expiration is no longer recommended.

You should force a user to change their password if you have reason to believe that it is weak or has been exposed in a data breach, but forcing arbitrary password changes is frustrating for users and does not meaningfully improve security.

Nevertheless, PingDirectory Server provides full support for password expiration. Use the following configuration properties to enable password expiration and customize its behavior:

`max-password-age`

The maximum length of time that a user can continue using the same password. If this is configured with a value of zero seconds (which is the default), then password expiration is disabled.

`password-expiration-warning-interval`

The length of time before an upcoming password expiration that the server should start warning about that expiration. A value of zero seconds disables the warning. By default, the server starts warning about an upcoming expiration five days in advance.

`expire-passwords-without-warning`

Indicates whether the server should allow a user's password to expire even if they have not been warned about an upcoming expiration. If this is false (which is the default), then the server ensures that the user receives at least one warning about the upcoming expiration, even if the expiration time has already passed. Once it has issued the warning, the server grants the user the full duration of the password expiration warning interval before the password actually expires.

`allow-expired-password-changes`

Indicates whether the server allows a user to change their password even after it has expired. This is false by default, and an administrator is required to reset the user's password before the account becomes usable again. However, if this is changed to true, then the user is allowed to use the password modify extended operation over an unauthenticated connection, providing their current password in addition to the desired new password.

`grace-login-count`

The maximum number of grace logins that the server grants to a user. A grace login allows a user to authenticate with an expired user, but only for the purpose of changing their password. Any other operations that they attempt are rejected. By default, the server does not allow any grace logins.

The server offers a pair of response controls that are related to password expiration, both of which are described in *draft-vchu-ldap-pwd-policy*:

- The password expiring response control is included in the response to a successful bind operation in cases where the user's password is about to expire.
- The password expired response control is included in the response to a bind operation in cases where the user's password has expired. If the bind response includes a result code of success, then the user is permitted to change their password, but is not be allowed to do anything else until they have done that. If the bind response has a non-success result code, then an administrative password reset is required to restore access to the user's account.

> ⓘ **Note:**
>
> If the user includes the password policy request control, as described in *draft-behera-ldap-password-policy* in the bind request, then the server includes the password policy response control in the bind response instead of the password expiring or password expired control. The password policy response control value can be used to indicate whether the user's password is expired or is about to expire, so the additional control is not necessary.

See the `config/sample-dsconfig-batch-files/enable-password-expiration.dsconfig` batch file for more information about enabling password expiration.

Failure lockout

PingDirectory Server provides the ability to lock a user's account after too many consecutive failed authentication attempts.

The lockout can be temporary, automatically expiring after a specified length of time, or permanent. In either case, an administrative password reset can be used to immediately restore access to the account.

However, actually locking a user's account after too many failed authentication attempts can be problematic because it has the potential to block legitimate access to the server. If the password policy is configured with a lockout failure count value that is too low, then it is possible for a user to legitimately mistype their password enough times in a row to trigger the lockout. However, the bigger risk is that an attacker could purposefully lock a user's account by intentionally making repeated failed bind attempts, denying the legitimate user access to their own account. To help protect against this, PingDirectory Server offers support for alternative failure lockout actions. Rather than locking the account, the server can be configured to delay bind responses as a rate-limiting mechanism, or it can be configured to generate an account status notification to alert administrators to the problem or invoke custom processing.

The password policy configuration offers the following properties related to failure lockout:

`lockout-failure-count`

The number of consecutive authentication failures required to trigger the failure lockout action. This is zero by default to indicate that failure lockout is disabled, but if it is set to a nonzero value, then the appropriate failure lockout action is taken after that number of consecutive failed attempts. If the user binds successfully before the failure count is reached, then the record of failed attempts is cleared. The set of failed attempts is also cleared by an administrative password reset.

`lockout-duration`

The length of time that the failure lockout should be in effect. A value of zero seconds (which is the default) indicates that the lockout should be permanent, and that an administrative password reset is required to restore access to the user's account. A value that is greater than zero indicates that the account is automatically unlocked after that length of time, although an administrative password reset can be used to restore access before that lockout period has expired.

`lockout-failure-expiration-interval`

The maximum length of time that the server should preserve information about a failed authentication attempt even if there is no intervening successful attempt or password reset. The default value of zero seconds indicates that failed attempts never expire, and they are preserved until enough failed attempts are accumulated to lock an account, or until a successful bind or an administrative reset clears the record of failed attempts.

`ignore-duplicate-password-failures`

Indicates that repeated failed authentication attempts that try the same wrong password should just be considered a single wrong attempt. If this is true (which is the default), then a client that repeatedly tries to bind with the same wrong password only accumulates a single failed attempt toward account lockout, regardless of the number of bind failures that actually occur. This is a

safeguard to prevent issues that might arise if an account's password is changed but there are applications still configured to use the old password. Using this option does not constitute an additional security risk because someone attempting an online guessing attack will try different passwords rather than repeatedly trying the same wrong password.

`failure-lockout-action`

Specifies the action that the server should take if the necessary number of failed attempts is reached.

See the `config/sample-dsconfig-batch-files/enable-failure-lockout.dsconfig` batch file for more information about enabling failure lockout.

Alternative failure lockout actions
As stated earlier, you can configure the server to take a variety of actions if the failure lockout count is reached.

Lock account

As its name implies, the lock account failure lockout action can be used to lock a user's account. Any attempts to authenticate are rejected while the lockout is active, regardless of whether the correct credentials were provided. If a lockout-duration is configured in the password policy, then the lockout automatically expires after that length of time. Otherwise, it persists until an administrative password reset.

This lockout failure action does not offer any configurable properties.

Delay bind response

The delay bind response lockout failure action can be used to delay the response to bind attempts after the lockout failure count has been reached. Once the threshold has been reached, then subsequent bind attempts are delayed until the user successfully authenticates or until the password is reset.

> ⓘ **Note:**
>
> If the server has delayed the response to any failed attempts, then the response to the next successful attempt will also be delayed. This helps prevent clients from using the presence of a delay as an indication of whether the password was correct.

Without this additional delay, an attacker could terminate the connection and establish a new one to make another attempt as soon as they detect the delay, rather than having to wait for the full duration of the delay.

The delay bind response failure lockout action offers the following configuration properties:

`delay`

The duration to use when delaying the response to the failed bind attempt. By default, a delay of one second is used.

`allow-blocking-delay`

Indicates whether the server should delay the bind response even doing so could block the thread being used to process the operation. This option applies to operations requested by non-LDAP clients. When delaying the response to an LDAP bind, the server is able to do so in a way that does not tie up any threads that could be used for processing operations. For other clients, like those using HTTP to communicate, the only option that the server has for performing the delay is to sleep on the thread that is processing the operation, which makes that thread unavailable for processing other requests until it's done sleeping. This property is set to false by default, and if you want to delay the response to HTTP clients, then you should make sure to adjust the number of HTTP

request handler threads to reduce the potential that all of them could be tied up sleeping on failed bind attempts.

`generate-account-status-notification`

Indicates whether the server should generate an account status notification if it delays a bind response. The account status notification uses a notification type of either `account-temporarily-locked` or `account-permanently-locked`, depending on whether the password policy is configured with a lockout duration. By default, no notification is generated.

No operation

The no operation failure lockout action does not take any client-visible action in response to reaching the lockout failure count. It offers a single configuration property:

`generate-account-status-notification`

Indicates whether the server should generate an account status notification, of type `account-temporarily-locked` or `account-permanently-locked`, if the failure lockout count is reached. This can potentially be used to notify administrators or optionally take some custom action using an account status notification handler created with the UnboundID Server SDK in the event of a failure lockout.

Sign on history tracking and idle account lockout

You can configure PingDirectory Server to maintain a record of previous authentication attempts. It can also lock an account if it has been too long since the user last successfully authenticated.

Recent sign on history
PingDirectory Server can maintain a history of recent successful and failed sign on attempts.

If enabled, it maintains the following information about each recorded attempt:

- A Boolean value indicating whether the attempt was successful
- A timestamp, formatted in the ISO 8601 format described in *RFC 3339*, indicating when the attempt occurred
- The name of the authentication method that was attempted (for example, “simple” or “SASL PLAIN”)
- The IP address of the client that made the attempt, if available
- A general reason that the authentication attempt failed for failed attempts
- An optional additional attempt count that can be used to indicate how many other attempts with the same properties (successful, authentication method, client IP address, and failure reason) occurred on the same date

Enabling recent login history

The following password policy configuration properties are used to manage recent login history tracking:

`maximum-recent-login-history-successful-authentication-count`

The maximum number of successful attempts to maintain in the recent login history.

`maximum-recent-login-history-successful-authentication-duration`

The maximum length of time to retain successful login attempts in the recent login history.

`maximum-recent-login-history-failed-authentication-count`

The maximum number of failed attempts to maintain in the recent login history.

`maximum-recent-login-history-failed-authentication-duration`

The maximum length of time to retain failed attempts in the recent login history.

recent-login-history-similar-attempt-behavior

The behavior to exhibit for clients with multiple similar attempts (with the same values for the successful, authentication method, client IP address, and failure reason fields) on the same date (within the UTC time zone). Allowed values include:

- `collapse-similar-attempts-on-the-same-date` — Indicates that multiple similar attempts should be collapsed into a single record. The timestamp of that record reflects the most recent attempt on that date, and the additional attempt count reflects the number of additional similar attempts that were collapsed. This is the default behavior.
- `maintain-every-attempt` — Indicates that the server should not collapse multiple similar attempts and that each attempt is maintained as a separate record in the recent login history. Clients that authenticate multiple times per day can have multiple records per day.
- `update-at-most-once-per-day` — Indicates that the server should not collapse multiple similar attempts and that only the first such attempt on any given day is recorded. This can reduce the number of writes required to maintain the recent login history.

If either the `maximum-recent-login-history-successful-authentication-count` or the `maximum-recent-login-history-successful-authentication-duration` properties has a value, then the server maintains a history of recent successful attempts. If both properties are configured, then the server can purge information about successful attempts that match the criteria for either. This is useful, for example, if you usually want to keep records based on a duration, but want to add protection against the history growing too large from an excessive number of records created within that duration.

Similarly, the server maintains a record of recent failed authentication attempts if either or both the `maximum-recent-login-history-failed-authentication-count` and `maximum-recent-login-history-failed-authentication-duration` properties are configured. It is possible to maintain a record of successful attempts without a record of failed attempts, to maintain a record of failed attempts without successful attempts, or to maintain a record of both successful and failed attempts. By default, no recent login history is maintained.

> ⓘ **Note:**
>
> If the `maximum-recent-login-history-successful-authentication-duration` and `maximum-recent-login-history-failed-authentication-duration` properties are used to maintain records of successful and failed attempts based on their duration, then it is possible for the server to retain records older than that duration if they are the most recent record of that type in the user's entry. That is, if the server is configured to maintain a history of successful logins, then the record of the most recent successful attempt will be retained even if it is older than the maximum duration for successful login attempts. The same is true if failed authentication attempts are to be maintained and a duration is configured.

See the `config/sample-dsconfig-batch-files/enable-recent-login-history.dsconfig` batch file for more information about configuring a recent login history.

Retrieving a user's recent login history

If the server is configured to maintain a recent login history for a user, then there are several ways that this history can be retrieved. They include:

- The client can include the get recent login history request control in the bind request. If the bind succeeds, then the server includes a corresponding response control in the bind result. The *UnboundID LDAP SDK for Java* provides support for these controls, and the `ldapsearch` and `ldapmodify` command-line tools both offer the `--getRecentLoginHistory` argument that can be used to retrieve the history from the command line.

- If the *ds-pwp-state-json* virtual attribute is enabled, then it might include a `recent-login-history` field whose value is a JSON object with information about recent successful and failed attempts for that user.
- The password policy state extended operation (or the `manage-account command-line` tool) can be used to retrieve the user's recent login history.

Last login time and IP address
Historically, the server has offered support for maintaining a record of the most recent successful attempt.

This is not as full-featured as the recent login history, but it might still be in use in legacy environments or when the full functionality of the login history is not needed.

The password policy configuration offers the following properties for maintaining a last login time and IP address:

`last-login-time-attribute`

The name of the attribute in which the last login time should be written. The server has reserved the `ds-pwp-last-login-time` operational attribute type for this purpose, and this value should not be changed.

`last-login-time-format`

The format in which the server should record the last login time. If both this property and the `last-login-time-attribute` property are assigned values, then the server generates a last login time value after each successful authentication attempt and updates the user's entry if the generated value does not match the value that is currently stored in the entry. Values for this property can use any valid format string that is compatible with the *java.text.SimpleDateFormat* class, but we recommend using one of the following values:

- `yyyyMMdd` — Indicates that the server should maintain a last login time containing only the date. This should cause each user's last login time to be updated at most once per day.
- `yyyyMMddHHmmss'Z'` — Indicates that the server should maintain the last login time in generalized time format using precision to the nearest second. This should cause the last login time to be updated for each successful authentication, unless the user authenticates multiple times in the same second.
- `yyyyMMddHHmmss.SSS'Z'` — Indicates that the server should maintain the last login time in generalized time format using precision to the nearest millisecond. This should cause the last login time to be updated for each successful authentication, unless the user authenticates multiple times in the same millisecond

`previous-last-login-time-format`

An optional set of alternative formats in which last login time values might have previously been written. If you have changed the value of the `last-login-time-format` property, then you should update this property with any former values so that the server can decode values generated in one of those earlier formats.

`last-login-ip-address-attribute`

The name of the attribute that should be updated with the IP address of the client from which the user last authenticated. The server has reserved the `ds-pwp-last-login-ip-address` operational attribute type for this purpose, but the value is not set by default because doing so would enable this feature and update the user's entry with the client IP address for each successful authentication attempt (unless the client IP address matches the current value for that attribute, in which case no update is needed).

See the `config/sample-dsconfig-batch-files/enable-last-login-tracking-and-idle-lockout.dsconfig` batch file for more information about enabling last login time and IP address tracking.

Idle account lockout
You can configure PingDirectory Server to lock a user's account if it has been too long since they last authenticated.

If the password policy has been configured to either maintain a record of recent successful authentication attempts or to maintain a last login time, then the following configuration property is used to enable idle account lockout:

`idle-lockout-interval`

Specifies the length of time that must pass after the most recent successful authentication for a user's account to be locked. A value of zero seconds (which is the default) indicates that idle account lockout should be disabled.

If an account has been locked because it has been too long since the user last authenticated, then it can be unlocked with an administrative password reset.

The `config/sample-dsconfig-batch-files/enable-last-login-tracking-and-idle-lockout.dsconfig` batch file provides more information about idle account lockout.

Self password changes

A self password change occurs whenever a user changes their own password.

This can occur when a user changes their own password on a connection authenticated as that user or when the request used to change the password includes the user's current password. This includes all of the following:

- When an authenticated client uses either a regular modify operation or a password modify extended operation to change the password for the account they used to authenticate, and when no alternate authorization identity has been requested.
- When an authenticated client uses a regular modify operation or password modify extended operation to change the password for an account, and uses the intermediate client request control, proxied authorization request control, or SASL alternate authorization to request that the operation be processed under the authority of the user that owns the account whose password is being changed.
- When a client uses the password modify extended operation to provide the current password for the account whose password is being changed. If the current password is provided, then this is considered a self-change regardless of whether the underlying connection is authenticated or the authenticated identity of that connection.
- When a client uses a regular modify operation and includes the current password for the user whose password is being changed, even if that request is received on a connection authenticated as some other user.

The password policy configuration includes the following properties pertaining to a user's ability to change their own password:

`allow-user-password-changes`

Indicates whether users are allowed to change their own passwords. If this is true (which is the default), then any user with access control permission to update their own password is permitted to do so (as long as the server considers the password acceptable). If this is set to false, then users are not allowed to change their own password regardless of the access control permissions that have been granted to them.

`password-change-requires-current-password`

Indicates whether users are required to provide their current password when choosing a new password.

`allow-expired-password-changes`

Indicates whether users are allowed to change their own password if it has already expired. If this is set to true, then they can use the password modify extended operation over an unauthenticated connection with both the current and desired new password.

Requiring current passwords for self password changes

If the `password-change-requires-current-password` property is set to true, then users are required to provide their current password when choosing a new password.

There are two ways that they can do this; using the password modify extended operation or using a regular password modify operation.

This is straightforward when using the password modify extended operation because the request already includes a field for the user's current password. If password-change-requires-current-password is true and the user does not provide the current password, then the operation is rejected. The attempt also fails if the provided current password is incorrect, and that is true regardless of the value of the password-change-requires-current-password property.

When using a regular LDAP modify operation, a user's current password can be provided in a password change request by including two modifications in that request: one that deletes the current password and another that adds the new password.

```
dn: uid=jdoe,ou=People,dc=example,dc=com
changetype: modify
delete: userPassword
userPassword: oldPassword
-
add: userPassword
userPassword: newPassword
-
```

See the `config/sample-dsconfig-batch-files/require-current-password-when-changing-passwords.dsconfig` batch file for more information about requiring users to provide their current password when performing self password changes.

Administrative password reset

An administrative password reset occurs when one user changes the password for another user.

This requires the password-reset privilege, and it also requires access control permission to update the password attribute.

You can configure PingDirectory Server to require users to choose a new password after their password has been reset by an administrator. This can be accomplished through the following configuration properties:

`force-change-on-add`

Indicates whether a user is required to choose a new password the first time they authenticate after their account has been created. This is false by default.

`force-change-on-reset`

Indicates whether a user is required to choose a new password the first time they authenticate after their password has been reset by an administrator. This is false by default.

`max-password-reset-age`

The maximum length of time that a user has to change their password after an administrative reset before their account is locked.

If a user is forced to change their password, then the server allows that user to authenticate with the new password provided by an administrator, but the bind response includes the password expired response

control and a diagnostic message indicating that they must change their password. The server also rejects any operation attempted on that connection until the user has chosen a new password.

If a maximum password reset age is configured and the user does not choose a new password within that length of time after the password reset, then the account is locked and another password reset is required to restore access to it.

See the `config/sample-dsconfig-batch-files/configure-password-reset-constraints.dsconfig` batch file for more information about forcing users to change their password after an administrative reset.

Password generators

PingDirectory Server provides support for password generators that are used under a variety of conditions.

These conditions include:

- If a client uses the password modify extended request to perform a self password change or an administrative password reset but does not include a new password in that request. In this case, the server can use a password generator to create a new password for the user and return it to the client in the password modify extended response.
- If a client uses an add request to create a new entry and includes the request control with that add request. The server generates a new password for that entry and returns it in a response control included with the add response.
- If the client uses the generate password extended operation, which can be used to request that the server generate one or more suggested passwords for a user. The server generates the requested number of passwords and returns them in the extended response.
- If the client uses the deliver one-time password extended operation, which can be used to generate a one-time password for use in the UNBOUNDID-DELIVERED-OTP SASL bind request.
- If the client uses the deliver password reset token extended operation, which can be used to generate a password reset token that can be used as an alternative to the user's current password the password modify extended request.
- If the client uses the deliver single-use token extended operation, which can be used to generate a token that can be used in conjunction with the consume single-use token extended operation.

You can configure the deliver one-time password, deliver password reset token, and deliver single-use token extended operation handlers to explicitly state the password generator that the server should use when creating those tokens. For the other use cases above, the server uses the password generator that is associated with the user's password policy. This can be specified with the following configuration property:

`password-generator`

Specifies the password generator that should be used for requests that require the server.

Random password generator

The random password generator constructs a password using a specified format.

It offers the following configuration properties:

`password-character-set`

Defines the character sets that can be used when generating passwords. Each character set should consist of a name followed by a colon and the set of characters that set contains (for example, `alpha:abcdefghijklmnopqrstuvwxyz` or `numeric:0123456789`). Multiple character sets can be configured.

`password-format`

Specifies the format that should be used when generating passwords. This should be a comma-delimited list in which each item is the name of a character set followed by a colon and the number of characters to include from that set (for example, `alpha:3,numeric:2,alpha:3` indicates that

the generated password should consist of three alphabetic characters followed two numeric digits and three more alphabetic characters).

Passphrase password generator
The passphrase password generator attempts to construct strong, memorable passphrases by combining multiple randomly selected words from a given dictionary file.

This password generator offers the following configuration properties:

`dictionary-file`

The path to the file containing the words to use when generating passwords. By default, the server includes a `passphrase-wordlist.txt` file with a large number of non-offensive English-langauge words.

`minimum-password-characters`

The minimum number of characters that should be included in the generated passphrase. The passphrase includes enough words to ensure that the minimum character count is reached. By default, generated passphrases include a minimum of 20 characters.

`minimum-password-words`

The minimum number of words that should be included in the generated passphrase. By default, passphrases contain at least four words.

`capitalize-words`

Indicates whether the first letter of each word should be capitalized, which can help make it easier to identify the words contained in the passphrase. By default, each word is capitalized. Generated passphrases are case-sensitive.

Third-party password generator
Use the UnboundID Server SDK to create custom password generator implementations.

Password retirement

When a user changes their own password, or when an administrator resets the password for another user, that password change takes place immediately, and the new password should be used for subsequent authentication attempts.

By default, the former password is also immediately removed from the user's entry and can no longer be used to authenticate. However, PingDirectory Server supports a feature called password retirement in which the user's former password can continue to be valid for a limited period of time.

Password retirement is especially useful for accounts used to authenticate applications, and particularly for applications that run on multiple systems. After the account's password is changed in the Directory Server, the application can continue trying to use the old password until it can be updated with the new one. Similarly, the application cannot be configured to use the new password before it is changed in the Directory Server because attempts to authenticate with that new password before the account has been updated will also fail. With password retirement, the password can be changed in the PingDirectory Server in a way that allows either the new password or the former password to be accepted. This provides a window of time in which the application instances can be updated with the new password.

The following password policy configuration properties can be used to configure password retirement:

`password-retirement-behavior`

The password retirement behavior that the server should exhibit. By default, passwords are not retired. However, any number of the following values can be used to enable retirement functionality:

- `retire-on-self-change` — Indicates that self password changes should automatically cause the user's former password to be retired unless the request includes the purge password request control.

- `retire-on-administrative-reset` — Indicates that administrative password resets should cause the user's former password to be retired unless the request includes the purge password request control.
- `retire-on-request-with-control` — Indicates that the client can use the retire password request control to indicate that the user's current password should be retired.

`max-retired-password-age`

The maximum length of time that the retired password should be considered valid. By default, this is set to one day.

The PingDirectory Server also provides support for two request controls that can be used to customize password retirement behavior. The retire password request control can be used to explicitly indicate that the user's current password should be retired, even if the server would otherwise purge it. This control is only allowed if the `password-retirement-behavior` property includes a value of `retire-on-request-with-control`. The purge password request control can be used to explicitly indicate that the user's current password should be removed from the server even if it would have otherwise retired it for example, because of the `retire-on-self-change` or `retire-on-administrative-reset` property.

See the `config/sample-dsconfig-batch-files/enable-retired-passwords.dsconfig` batch file for more information about configuring password retirement.

Password reset tokens

If a user loses access to their account for some reason, such as if they forget their password or let it expire, then they must have their password reset by an administrator to regain access to it.

This can be costly for the organization because they might have to employ additional help desk staff, and it can be frustrating for users who might have to wait on someone to become available to help them.

Many organizations have attempted to deal with this by automating the password reset process: their application can provide an "I forgot my password" link option that identifies the user and sends them a new password that can be used to authenticate. PingDirectory Server provides support for password reset tokens that can be used to provide much of the heavy lifting.

Much like delivered one-time passwords, a password reset token is a single-use password that is generated by the server and delivered to the user through an out-of-band mechanism like email or SMS. However, rather than allowing the user to authenticate, a password reset token can only be used in the current password field of a password modify extended request so that they can choose a new password.

After an application has obtained whatever it considers sufficient evidence of the user's identity, such as by asking them for their username, email address, or phone number, it can use the deliver password reset token extended request to have the server generate a single-use password reset token and deliver it to the user through some out-of-band mechanism, and then it can allow the user to enter that token and their desired new password to send a password modify extended request to actually change the password.

The following password policy configuration property can be used to indicate the conditions under which a password reset token can be used:

allowed-password-reset-token-use-condition

The set of conditions under which a password reset token can be used. Allowed values include:

- `account-usable` — Indicates that a password reset token can be used if the user's account is in a usable state and would permit them to authenticate if they provided the correct credentials.
- `account-locked-due-to-failures` — Indicates that a password reset token can be used to recover access to an account that has been locked after too many failed authentication attempts.
- `account-locked-due-to-idle-time-limit` — Indicates that a password reset token can be used to recover access to an account that has been locked because it has been too long since the user last authenticated.

- `account-locked-due-to-admin-reset-timeout` — Indicates that a password reset token can be used to recover access to an account that has been locked because they failed to choose a new password in a timely manner after an administrative password reset.
- `account-locked-due-to-validation-failure` — Indicates that a password reset token can be used to recover access to an account that has been locked because their password failed to satisfy one or more bind password validators.
- `password-expired` — Indicates that a password reset token can be used to recover access to an account with an expired password.

In addition to the password policy configuration, you must configure one or more one-time password delivery mechanisms in the server and create an instance of the deliver password reset token extended operation handler. The server offers out-of-box support delivering one-time passwords as SMS (using the Twilio service) or email messages, and the Server SDK provides support for developing custom delivery mechanisms.

The deliver password reset tokens extended operation handler offers the following configuration options:

`password-generator`

The password generator that should be used to create the password reset tokens.

`default-token-delivery-mechanism`

An ordered list of one-time password delivery mechanisms that should be tried if the extended request does not indicate which methods to attempt.

`password-reset-token-validity-duration`

The length of time that password reset tokens should be valid. They are valid for five minutes by default.

See the `config/sample-dsconfig-batch-files/support-password-reset-tokens.dsconfig` batch file for more information about configuring the server to support password reset tokens.

Account status notifications

Account status notifications are used to notify users (or administrators) about events that affect their accounts or potentially to invoke custom code if such events occur.

The types of events that can generate account status notifications include:

- The account has been locked after too many failed authentication attempts.
- A bind attempt failed because it has been too long since the user last authenticated.
- A bind attempt failed because the user did not choose a new password in a timely manner after an administrative reset.
- A bind attempt failed because the password was rejected by one or more bind password validators.
- The account has been unlocked by an administrator.
- The account has been disabled or enabled by an administrator.
- A bind attempt failed because the account is not yet active or has expired.
- A bind attempt failed because the password is expired.
- The user's password is about to expire.
- The user has changed their own password.
- The user's password has been reset by an administrator.
- The user's account has been created with an add request that matches a defined set of criteria.
- The user's account has been updated with a modify or modify distinguished name (DN) request that matches a defined set of criteria.

The following password policy configuration property can be used to configure one or more account status notification handlers for use with that policy:

account-status-notification-handler

The set of account status notification handlers that should be invoked for users associated with the password policy.

The server offers support for a few types of account status notification handlers by default, including:

- A multi-part email account status notification handler that can generate elaborate email messages (containing plain-text and HTML-formatted body text and an optional set of attachments) from customizable templates.
- A legacy SMTP account status notification handler that can generate simple plain-text email messages.
- An error log account status notification handler that can record a message in the server's error log when such events occur.

You can also use the UnboundID Server SDK to create custom account status notification handlers if desired.

See the `config/sample-dsconfig-batch-files/enable-email-account-status-notifications.dsconfig` batch file for more information about configuring the multi-part email account status notification handler, and see the `config/account-status-notification-email-templates/README.txt` file for a detailed overview of the options that are available for customizing the email templates.

Other password policy configuration properties

The password policy configuration also provides additional configuration properties that don't fall into any of the previously discussed categories.

They include:

password-attribute

Specifies the attribute used to hold the password in the user's account. This is userPassword by default, but it can also be set to authPassword if you want to use the authentication password schema described in *RFC 3112*.

require-secure-authentication

Indicates whether users associated with this policy are required to authenticate in a secure manner. This is false by default, but we strongly recommend setting it to true.

requires-secure-password-changes

Indicates whether users associated with this policy are required to change their password in a secure manner. This is false by default, but we strongly recommend setting it to true.

allow-multiple-password-values

Indicates whether accounts are allowed to have multiple different passwords. Although this is technically allowed by LDAP specifications, it is strongly discouraged because it can be abused to allow a user to exempt themselves from certain password policy constraints like password expiration. If a user needs different passwords for different purposes, then we recommend creating separate accounts for that user.

require-change-by-time

Can be used to require that all users associated with the password policy change their password by a specified time. For example, this can be used to require all users to change their passwords after a data breach.

Managing password policy state

Because PingDirectory Server offers a wide range of password policy functionality, it also has the ability to maintain a wide range of state information that corresponds to those features. It provides several ways to access and manipulate this state.

Externally modifiable user attributes
A limited set of operational attributes can be directly manipulated (for example, through LDAP add or modify operations) to manage certain aspects of a user's password policy state.

They include:

`ds-pwp-password-policy-dn`

The distinguished name (DN) of the password policy that governs the user. If this is not present in the user's entry (as either a real or virtual attribute), then the user is subject to the server's default password policy.

`ds-pwp-account-disabled`

Indicates whether a user's account should be administratively disabled. If this attribute is present with a value of true, then the account is disabled. If this attribute is present with a value of false, or if the attribute is absent, then the account is enabled.

`ds-pwp-account-activation-time`

Specifies the time at which a user's account becomes active. Attempts to authenticate as the user (or use the account as an alternate authorization identity) fails before this time.

`ds-pwp-account-expiration-time`

Specifies the time at which a user's account will expire. Attempts to authenticate as the user (or use the account as an alternate authorization identity) fails after this time.

`ds-auth-totp-shared-secret`

A shared secret that can be used to generate time-based one-time passwords in conjunction with the UNBOUNDID-TOTP SASL mechanism. Although this attribute can be manually updated, we recommend using the generate Time-based One-time Password (TOTP) shared secret extended operation for generating a shared secret and storing it in the user's entry.

`ds-auth-preferred-otp-delivery-mechanism`

The public identifier of a YubiKey device that can be used to generate one-time passwords for use in conjunction with the UNBOUNDID-YUBIKEY-OTP SASL mechanism. Although this attribute can be manually updated, we recommend using the registered YubiKey OTP device extended operation.

Administrative password reset
In most cases, if a user's account is in an unusable state, an administrative password reset can restore access.

This includes:

- If a user has forgotten their password
- If the account has been locked after too many failed authentication attempts
- If the account has been locked because it has been too long since the user last authenticated
- If the account has been locked because the user failed to authenticate in a timely manner after an administrative reset
- If the account has been locked because the user attempted to bind with a password that failed validation
- If the user's password has expired

An administrative password reset does not restore access to an account under the following circumstances:

- If the account has been administratively disabled
- If the account has an activation time that is in the future

  If the account has an expiration time, not to be confused with the password expiration time, which is in the past

The password policy state extended operation and the manage-account tool
PingDirectory Server supports a proprietary *password policy state extended operation* that can retrieve and manipulate virtually any kind of password policy state information in a user's entry.

This includes:

- Retrieving the DN of the password policy that governs the user
- Retrieving a flag that indicates whether the server considers the account usable
- Retrieving a set of error, warning, and notice conditions that can affect the account's usability
- Determining whether the account has a static password

  Retrieving and updating the flag indicating whether an account is disabled
- Retrieving and updating the account's activation and expiration times
- Retrieving and updating the account's password changed time
- Determining whether the user's password is expired
- Retrieving the account's password expiration time, which is computed from the password changed time
- Retrieving and updating the account's password expiration warned time
- Retrieving and updating the set of grace login use times
- Retrieving and updating the record of failed authentication attempts
- Retrieving and overriding a failure-based account lockout
- Retrieving the time that an account was failure locked
- Retrieving and updating an account's last login time
- Retrieving and updating an account's last login IP address
- Retrieving and clearing an account's recent login history
- Retrieving the length of time until an upcoming idle lockout
- Retrieving and updating the account's "must change password" flag
- Determining whether an account is reset locked
- Retrieving the length of time until an password reset lockout
- Retrieving the number of passwords in the user's history and clearing the history
- Determining whether a user has a retired password and purging the retired password
- Retrieving the set of SASL mechanisms that are available to the user
- Retrieving the set of one-time passcode (OTP) delivery mechanisms that are available to the user
- Determining whether the user has any TOTP shared secrets
- Registering and deregistering TOTP shared secrets
- Determining whether the user has any registered YubiKey OTP devices
- Registering and deregistering YubiKey OTP devices
- Retrieving and updating the time that bind password validation was last performed for the user
- Retrieving and clearing password validation lockout

The server also includes a manage-account tool that provides command-line access to the functionality of the password policy state extended operation.

The ds-pwp-state-json and ds-pwp-modifiable-state-json operational attributes
The PingDirectory Server provides a *ds-pwp-state-json* operational attribute whose value is a JSON object that provides read-only access to a wide variety of password policy state information and related password policy configuration.

This includes:

- The distinguished name (DN) of the password policy that governs the user

- Whether the account is usable, and information about any usability errors, warnings, and notices
- Whether the account has a static password
- The password changed time
- Whether the account is disabled
- Whether the account has an activation time or expiration time, and their relation to the current time
- Whether the user's password is expired and when it expires
- Whether the user has been warned about an upcoming password expiration
- Whether the account is locked as a result of too many failed authentication attempts
- The last login time, last login IP address, and recent login history
- Whether the account is locked because it has been too long since they last authenticated
- Whether the user must change their password
- Whether the user's account is locked because they failed to choose a new password after an administrative reset
- The number of passwords in the user's password history
- Whether the user is within the minimum password age
- Information about grace login uses
- Information about whether the user has a retired password and how long it will be valid
- Whether the server will require secure authentication or secure password changes
- A list of SASL mechanisms available to the user
- A list of onte-time passcode (OTP) delivery mechanisms available to the user
- Whether the account is locked as a result of bind validation failure

It also provides a `ds-pwp-modifiable-state-json` operational attribute that can be used to manipulate a limited set of password policy state information, including:

- The user's password changed time
- Whether the user's account is disabled
- The user's account activation time
- The user's account expiration time
- Whether the user's account is locked as a result of too many failed authentication attempts
- The time the user was warned about an upcoming password expiration
- Whether the user must change their password

The value of the `ds-pwp-modifiable-state-json` attribute is a JSON object, and it can be updated in a replace modification. Any of the fields that it contains can be included in the replace modification to manipulate the corresponding portions of the user's password policy state. Any fields omitted from the JSON object are not updated.

The password update behavior control

PingDirectory Server provides support for a *password update behavior request control* that allows an authorized user with the `password-reset` privilege and access control permission to use the control to override the server's default behavior when updating the password.

It can be included in an add request, modify request, or password modify extended request.

The options that it provides include:

- Force the server to treat the operation as a self change or an administrative reset.
- Force the server to accept a pre-encoded password.
- Force the server to skip validation for the password.
- Force the server to ignore the user's password history.
- Force the server to ignore the user's minimum password age.
- Force the server to use a specified password storage scheme.
- Force the server to set or clear the "must change password" flag.

The retire password and purge password controls

Including the retire password and purge password controls in a modify request or a password modify extended request affect a user's current password and a user's ability to authenticate.

The *retire password request control* can be included in a modify request or a password modify extended request to indicate that the user's current password should be retired so that it can continue to be used to authenticate to the account, as an alternative to the new password, for a limited period of time.

The *purge password request control* can be used to indicate that the user's current password should be purged from the entry even if it would have otherwise been retired based on the password policy configuration.

Authentication-related controls and extended operations

PingDirectory Server also provides support for a number of additional controls and extended operations that can be used when authenticating or interacting with password policy-related operations.

The authorization identity request control

The authorization identity request control is described in *RFC 3829* and can be included in a bind request to indicate that the server should include the resulting authorization identity in the successful bind response.

In PingDirectory Server, this authorization identity is always in the form of a distinguished name (DN), prefixed by `dn:` (for example, `dn:uid=jdoe,ou=People,dc=example,dc=com`).

This control is useful to determine the DN of the authenticated user entry, especially when the bind request does not identify the user by a DN, such as if the client was identified by a username, a Kerberos principal, a client certificate, or an OAuth access token.

The get authorization entry request control

While the authorization identity request control can be helpful, it only provides the distinguished name (DN) of the authenticated user and the specification does not even require that.

PingDirectory Server offers a more useful alternative in the form of the proprietary *get authorization entry request control*.

This control can also be included in a bind request, and if the bind is successful, then the server can return a corresponding response control with the DN and a requested set of attributes from the authenticated user's entry. If the bind request also used an alternate authorization identity, then the response control can also include information about that user.

The "Who am I?" extended request

The "Who am I?" extended request is defined in *RFC 4532*. It behaves much like the authorization identity request control in that it returns the authorization identity associated with the connection and it should always be in `dn:` form in PingDirectory Server, but this request can be issued at any time on a connection to retrieve the current authorization identity for that connection.

The account usable control

Include the *account usable request control* in a search request to indicate that matching entries should include a corresponding response control with a minimal set of information about whether the server considers the associated account to be usable.

This includes:

- Whether the account is usable
- The length of time until the user's password expires
- Whether the account is inactive
- Whether the user must change their password
- Whether the password is expired
- The number of remaining grace logins

- The length of time until the account is unlocked

This control is maintained for compatibility with a legacy system. While it is still useful, it is not updated to support new features.

The password policy control
PingDirectory Server supports the password policy request control, as described in *draft-behera-ldap-password-policy-10*.

This control can be included in add, bind, compare, modify, and password modify extended requests to obtain information about the associated user's password policy state. This includes:

- The length of time until the user's password expires
- The number of remaining grace logins
- Whether the password is expired
- Whether the account is locked
- Whether the user must change their password
- Whether an update attempt failed because the user is not allowed to change their password
- Whether an update attempt failed because the user is required to provide their current password
- Whether an operation failed because the password is considered too weak
- Whether the proposed password is too short
- Whether the proposed password already exists in the user's password history
- Whether a user cannot change their password because there has not been enough time since the previous password change

Because this control is based on a public specification, its format is fixed and it is not updated to support additional features.

The password expiring and password expired controls
PingDirectory Server supports the password expiring and password expired controls, as described in *draft-vchu-ldap-pwd-policy-00*.

The password expiring control can be included in the response to a successful bind attempt to indicate that the user's password is about to expire. Its value indicates the length of time until the password actually expires.

The password expired control can be included in the response to a successful or failed bind attempt to indicate that the user's password has expired and must be changed. If the bind operation was successful, then it means that the user must change their password before they are allowed to request any other operations. If the bind operation failed, then it means that the password must be reset before the user can access their account.

The get password policy state issues control
By default, PingDirectory Server returns a minimal amount of information in the response to a failed bind attempt. This is intentional because revealing too much can give an attacker useful information that could allow them to improve their tactics.

Nevertheless, it is useful in some circumstances to provide an application with a way to obtain information about the reason for a failed authentication attempt. As such, PingDirectory Server offers a *get password policy state issues request control* that can be included in a bind request to indicate that the server should include a control in the bind response with information about any error, warning, or notice conditions in the user's password policy state that might currently or soon interfere with their ability to authenticate. If the bind attempt fails, then it might also include specific information about the reason for that failure.

To prevent this control from being misused, PingDirectory Server only allows it to be requested under a limited set of circumstances:

- The bind request must be issued on a connection that is currently authenticated as a user with the `permit-get-password-policy-state-issues` privilege.

- The requester must have access control permission to use the get password policy state issues request control.

The bind request must also include the *retain identity request control* in the bind request.

The get password quality requirements extended operation
Use the proprietary *get password quality requirements extended operation* to request that the Directory Server provide a list of the requirements that the server enforces when setting a password.

The request control allows you to indicate the context in which the password is provided (for example, whether it's an add request, a self password change, or an administrative password reset) because the requirements can vary in different circumstances.

The extended result includes a list of the requirements the server will impose, including a human readable description for each and an optional name and set of properties that can allow for some client-side validation. The result might also indicate whether the user is required to provide their current password, whether they are required to choose a new password after the add or administrative reset, and how long a new password is valid before it expires.

The password validation details control
PingDirectory Server provides support for a *password validation details request control* that can be included in an add request, modify request, or password modify extended request.

It indicates that the server should return a corresponding response control with a list of each of the requirements that the password must satisfy and an indication as to whether the proposed password satisfied.

The generate password request control
You can include the proprietary *generate password request control* in an add request to indicate that the server should automatically generate a password for the user.

The add response includes a corresponding response control that contains the generated password, along with an indication as to whether the user is required to choose a new password and how long the generated password is valid.

The password is generated using the password generator defined in the password policy that governs the new entry. Although the server might attempt to re-generate the password multiple times if previous attempts do not satisfy the configured set of password validators for the user, it might end up setting a password that does not pass validation. We recommend configuring the password generator to ensure that it is capable of generating passwords of acceptable strength.

The generate password extended operation
The proprietary *generate password extended operation* is used to request that PingDirectory Server generate one or more suggested passwords and return them to the client.

This is intended for use by applications that allow a user to change their password or allow an administrator to reset user passwords and want to allow the server to suggest a number of strong options that the user can choose from or use as examples when choosing their own password.

As with the generate password request control, this extended operation relies on a password generator to create the passwords. While it might optionally try multiple times to generate passwords that pass validation, it can ultimately return passwords that do not satisfy all of the configured password validators. For each generated password included in the extended response, the server indicates whether that password passed validation, and if not, it might include a list of errors that explain the reasons it was not considered acceptable.

Access control

The server's access control subsystem provides a way to examine each request that a client issues to determine whether it should be allowed.

It can also examine each search result entry to determine whether the client should be permitted to retrieve it at all, and if so, which attributes or even attribute values should be permitted.

The server's access control policy is constructed from a set of access control instructions (ACIs), also called access control rules. ACIs can be defined in user data in the ACI operational attribute, and they can also be defined in the configuration in the `global-aci` property in the access control handler configuration.

The server's access control policy denies all access by default. Unless there is an ACI that allows something, then no user who is subject to access control is permitted to perform the requested operation or retrieve the specified data. It is also possible to explicitly deny access to something, which overrides any permission that would have otherwise granted access to it.

ACI syntax

The access control instruction (ACI) syntax that PingDirectory Server uses is similar to that used by other types of directory servers and is designed to make it easy to migrate data containing access control rules from other servers.

Each ACI has the following format:

1. A set of one or more targets. The targets specify the data, attributes and entries, to which the ACI applies. Each ACI target is enclosed in parentheses and has the form (`name="value"`). Defined target names include:
   - `targetattr` — Provides a list of attributes to which the ACI applies.
   - `target` — Provides an LDAP URL whose distinguished name (DN) identifies the base of the subtree to which the ACI applies.
   - `targetscope` — Specifies the scope, relative to the target base DN, to which the ACI applies.
   - `targetfilter` — Specifies a filter used to restrict the set of entries to which the ACI applies within the scope.
   - `targattrfilters` — Provides criteria for identifying the values within an attribute to which the ACI applies.
   - `targetcontrol` — Provides a list of request control OIDs to which the ACI applies.
   - `extop` — Provides a list of extended request OIDs to which the ACI applies.
2. An opening parenthesis.
3. The string `version 3.0` to indicate the ACI syntax version. PingDirectory Server only supports 3.0.
4. A semicolon followed by a space.
5. The keyword `acl` followed by a space and a description for the access control instruction surrounded by quotation marks, such as `acl"Allow users to update their own entries"`.
6. A semicolon followed by a space.

7. The keyword `allow` or `deny` followed by a comma-delimited list of rights enclosed in parentheses. For example, `allow` (`read`,`search`,`compare`). Available rights include:
   - `read` — Indicates that the ACI grants or denies permission to retrieve attributes in search result entries.
   - `search` — Indicates that the ACI grants or denies permission to use attributes in a search filter.
   - `compare` — Indicates that the ACI grants or denies permission to request a compare assertion against target attributes.
   - `write` — Indicates that the ACI grants or denies permission to update attributes within target entries.
   - `selfwrite` — Indicates that the ACI grants or denies permission to allow the requester to update attributes to add or remove their own DN.
   - `add` — Indicates that the ACI grants or denies permission to add entries.
   - `delete` — Indicates that the ACI grants or denies permission to delete entries.
   - `export` — Indicates that the ACI grants or denies permission to move an entry out from under its current parent.
   - `import` — Indicates that the ACI grants or denies permission to move an entry beneath its proposed new parent.
   - `all` — Indicates that the ACI grants or denies permission to all of the above rights. It does not include the `proxy` right.
   - `proxy` — Indicates that the ACI grants or denies permission for one user to request that an operation be authorized as a different user, such as using a proxied authorization request control or SASL alternate authorization.
8. A semicolon followed by a space.
9. The bind rule component for the ACI, which identifies the set of requesters to which the ACI applies. Multiple bind rules can be joined with the keywords `and` and `or`, and the keyword `not` can be used to negate the result of a rule. Available bind rules include:
   - `userdn` — Identifies the requester by DN or by a predefined keyword that is interpreted by the server.
   - `groupdn` — Identifies the requester by group membership.
   - `userattr` — Identifies the requester by their relation to the value of a specified attribute.
   - `authmethod` — Identifies the requester by the type of authentication that they used.
   - `ip` — Identifies the requester by the IP address of the client system.
   - `dns` — Identifies the requester by the resolved name of the client system.
   - `dayofweek` — Identifies the requester by the current day of the week.
   - `timeofday` — Identifies the requester by the current time of day.
   - `oauthscope` — Identifies the requester by the set of OAuth scopes that they have.
10. A semicolon followed by a closing parenthesis.

For example, the following ACI will allow a user to update their own password.

```
(targetattr="userPassword")(version 3.0; acl "Allow a user to update their
 own password"; allow (write) userdn="ldap:///self";)
```

ACI targets

ACI targets specify the set of data to which an ACI applies.

Each ACI target should be surrounded by parentheses and should have the form `(name="<value>")`.

An access control instruction can have multiple targets by simply placing them one right after another such as `(targetattr="userPassword")(target="ldap:///ou=People,dc=example,dc=com")(targetscope="sub")(targetfilter="(objectClass=person)")`. All of the ACI targets must appear at the beginning of the ACI, before.

`targetattr`

The `targetattr` ACI target is used to grant or deny access to a specified set of attributes. If multiple attributes are specified, then they should be separated by two consecutive vertical bars. For example, `(targetattr="givenName||sn||cn")` indicates that the ACI applies to the givenName, sn, and cn attributes.

The token `*` can be used as a shorthand notation to indicate all user attributes, but it does not include operational attributes. The token `+` can be used as a shorthand notation to indicate all operational attributes, but does not include user attributes. If you want an ACI to apply to all user attributes and all operational attributes, then you can use `(targetattr="*||+")`.

It is possible to use the `!=` operator to target all user attributes except a named set. For example, `(targetattr!="userPassword")` indicates that the ACI targets all userAttributes except `userPassword`. However, this is potentially dangerous because it is easy for two different ACIs using this construct to cancel each other out. For example, if one ACI includes `(targetattr!="userPassword")` and a second ACI includes `(targetattr!="socialSecurityNumber")`, then the net effect might be equivalent to `(targetattr="*")` because the first target implicitly includes the `socialSecurityNumber` attribute that you're trying to exclude with the second, while the second implicitly includes the `userPassword` attribute that you're trying to exclude with the first. When access to a specific attribute should not be allowed, it might be better to create an ACI that denies access to that attribute rather than one that grants all access to all attributes except that attribute.

`target`

The `target` ACI target is used to indicate that the ACI applies to entries in specified portions of the DIT. The value should be provided as an LDAP URL, but only the DN portion of that URL is used. For example, `(target="ldap:///dc=example,dc=com")` indicates that the ACI applies to entries at or below "`dc=example,dc=com`".

Each ACI cam have at most one target. If an ACI defined in the user data does not contain a target element, then the effective target for that ACI the DN of the entry in which that ACI is defined. If a global ACI does not contain a target element, then it applies to all entries in the server, including in backends containing non-user data, like the root DSE, configuration, changelog, schema, and monitor backends. The `target` element can only use the "`=`" operator. It cannot use the `!=` operator to apply to all portions of the DIT outside of the specified distinguished name (DN).

If an ACI defined in user data includes a target element, then the DN contained in that element must be at or below the DN of the entry that contains that ACI. For example, the `ou=People,dc=example,dc=com` entry cannot have an ACI that targets entries below `ou=Groups,dc=example,dc=com`.

If an ACI should not apply to all entries within the target subtree, then the `targetscope` and `targetfilter` elements can be used to narrow down the set of applicable entries.

`targetscope`

The `targetscope` ACI target is used to specify how much of the target subtree is covered by the ACI. These scopes exhibit the same behavior in ACIs as they do for search operations. Allowed scope values include:

`base`

Indicates that the ACI only applies to the entry specified by the target DN, and not to any of its subordinates.

`onelevel`

Indicates that the ACI only applies to entries that are immediate subordinates of the entry specified by the target DN. Neither the target entry, nor any entries that are more than one level below the target entry, are included.

`subtree`

Indicates that the ACI applies to the entry specified by the target DN, as well as all entries below it (to any depth).

`subordinate`

Indicates that the ACI applies to all entries below the entry specified by the target DN (to any depth), but does not include the target entry itself.

The `targetscope` element can only use the "=" operator. It cannot use the `!=` operator. If an ACI does not include a `targetscope` element, a default value of `subtree` is assumed.

`targetfilter`

The `targetfilter` ACI target is used to identify the set of entries in the target subtree to which the ACI applies based on their content. The value of this element is the string representation of an LDAP search filter, such as "`(targetfilter="(objectClass=person)")`".

The `targetfilter` element can only use the "=" operator; it cannot use the "`!=`" operator although you can use the "`!`" operator in the filter itself to negate its meaning. If an ACI does not include a `targetfilter` element, then all entries within the target subtree and scope are considered applicable.

`targattrfilters`

The `targattrfilters` ACI target is used to identify individual attribute values to which the ACI applies. This target is used for write operations, and it can restrict which attributes can be added to or removed from an entry.

The value of this element can have either one or two components. It can specify a set of attribute values that can be added to the entry (which must start with `add=`), a set of attribute values that can be removed from the entry (which must start with "`del=`"), or both. If both clauses are present, then they should be separated by a comma.

The content of each clause should be in the form of an attribute name followed by a colon and a search filter that identifies which values will be allowed for that attribute You can target multiple attributes in the same add or delete clause by separating them with a double ampersand `&&`.

For example, `(targattrfilters="add=description:(description=allowedAddDescription) && displayName:(displayName=allowedAddDisplayName), del=description:(description=allowedDeleteDescription)")` indicates that the ACI applies to any attempt to add a value of `allowedAddDescription` to the description attribute, an attempt to add a value of `allowedAddDisplayName` to the `displayName` attribute, or an attempt to remove a value of `allowedDeleteDescription` from the description attribute.

The filters can use a variety of logic for identifying which values are allowed. For example, you can use substring filters that include wildcards to match values that start with, end with, or contain a given substring. You can use a presence filter to indicate that any value is allowed. You can use a greater-or-equal or less-or-equal to indicate a range of allowed values. You can use AND and OR filters to combine multiple filters. You can use a NOT clause to negate a filter.

When processing an add operation, the server requires that the requester have permission to add all of the attribute values. When processing a delete operation, the server requires that the requester have permission to delete all of the attribute values. When processing a modify or modify DN operation, the server requires that the requester have permission to add all values introduced in the update and to delete all values removed in the update.

The `targattrfilters` element can only use the = operator; it cannot use the `!=` operator.

`targetcontrol`

The `targetcontrol` ACI target is used to indicate which request controls a requester is allowed to use. The value for this element is the OID for the desired request control, and multiple request control OIDs can be separated by a double vertical bar `||`. For example, `(targetcontrol="1.2.840.113556.1.4.473||2.16.840.1.113730.3.4.9")` is used to target requests that use either or both the server-side sort request control whose OID is "`1.2.840.113556.1.4.473`" and the virtual list view request control whose OID is `2.16.840.1.113730.3.4.9`.

Regardless of the type of control and what action the server can take for requests that contain it, the read right is used when determining whether to allow an operation containing the request control. Although it is technically allowed to include other rights in an ACI that uses the `targetcontrol` target, only the read right is used in making the determination.

The `targetcontrol` element can only use the `=` operator; it cannot use the `!=` operator.

`extop`

The `extop` ACI target is very similar to the `targetcontrol` target, except that it is used to indicate which extended requests a requester is allowed to use. The value can be either a single OID or a list of multiple OIDs separated by double vertical bars.

As with the `targetcontrol` element, only the read right is considered when determining whether to allow a client to request a given extended operation.

> ⓘ **Note:**
>
> The server might impose additional access control rights based on the function that the extended operation is to perform. For example, for a client to use the password modify extended operation, the server requires an ACI with an `extop` target containing the OID `1.3.6.1.4.1.4203.1.11.1`, but it also requires that the requester have permission to update the `userPassword` attribute or whatever password attribute has been configured in the user's password policy in the target user's entry.

ACI rights

The rights section of an ACI defines the permissions that are granted or denied to requesters identified by the bind rule for operations against the data specified by the target.

Every ACI must allow or deny one or more rights.

`read`

The `read` right covers access to attributes within search result entries. If a client does not have the `read` right for an attribute in a search result entry, then it is stripped out of the entry when it is returned to the client.

`search`

The `search` right covers permission to use attributes in a search filter. When performing a search (regardless of its scope), the requester must have `search` permission for all attributes in the filter.

If the requester has `search` permission for all attributes used in the filter, but only for a portion of the subtree used as the scope for the search, then only entries that reside in portions of the DIT where the search right is granted can be retrieved. For example, if a user has the search right for the `cn` attribute below `ou=People,dc=example,dc=com`, then a search based at `dc=example,dc=com` with a filter that contains the `cn` attribute only returns matching entries below `ou=People,dc=example,dc=com` even if there are other entries matching the filter below `dc=example,dc=com` but outside of `ou=People,dc=example,dc=com`.

`compare`

The `compare` right covers permission to perform a compare assertion for a specified set of attributes.

A compare assertion includes an entry DN, an attribute name, and an assertion value. If the specified entry has the given attribute with the provided assertion value, then the server returns a result of `compareTrue` (result code 6). If the entry does not have the indicated attribute value, then the server returns a result of `compareFalse` (result code 5). However, if the requester does not have permission to perform that compare assertion, then the server returns a result of `insufficientAccessRights` (result code 50).

`write`

The `write` right covers permission to update attributes in an entry. This includes modify operations, and it also includes modify DN operations that do not specify a newSuperior (that is, modify distinguished name (DN) operations that only attempt to rename an entry and do not attempt to move it beneath a new parent). This does not include adding new entries or deleting existing entries.

`selfwrite`

The `selfwrite` right is a limited subset of the `write` permission. It covers permission for a user to add their own DN to the set of values for specified attributes or for a user to remove their own DN from the set of values for those attributes. This is typically used to allow a user to add themselves to or remove themselves from static groups.

The `selfwrite` right should only be used for attributes that have a syntax of either distinguished name or name and optional UID. Attempts to use it for attributes with other syntaxes can fail or result in unexpected behavior.

`add`

The `add` right covers permission to add new entries to the server. The requester must have `add` or `write` permission for all attributes included in the entry to be added.

`delete`

The `delete` right covers permission to remove entries from the server. For the delete operation, the requester only needs to have the `delete` right for the target entry and not for individual attributes within the entry. However, the server enforces any `targattrfilters` restrictions for attribute values contained in the entry to be deleted. If a `targattrfilters` restriction is used to limit the set of values that the requester can delete, then they are only allowed to delete entries containing those values.

`export` and `import`

Although you might assume otherwise from their names, the `export` and `import` rights do not have any relation to exporting data to LDIF or importing data from LDIF. Instead, these rights cover permission to move entries within the DIT (using a modify DN operation that includes a newSuperior). The `export` right is required to move an entry out from under its current parent, and the `import` right is required to move the entry below its new parent.

These rights are not required to perform a modify DN operation that does not attempt to move the entry below a new parent. That is covered by the `write` right.

`all`

The `all` right is a shorthand notation that includes the capabilities of all of the other access control rights except the `proxy` right. Using the `all` right is equivalent to using `read`, `search`, `compare`, `write`, `selfwrite`, `add`, `delete`, `export`, and `import`.

`proxy`

The `proxy` right covers the ability to process an operation under the authority of an alternate authorization identity. This includes:

- Requests that include a proxied authorization request control
- Requests that include an intermediate client request control with a userIdentity
- SASL bind requests that request an alternate authorization identity

Because the ability to impersonate another use is a very sensitive operation, the requester must have the `proxied-auth` privilege for the operation to be allowed.

ACI bind rules
Bind rules are used to identify the set of requesters to which an ACI applies.

ACIs can have multiple bind rules to specify multiple conditions that can be satisfied.
Bind rules can be combined using either the `and` or the `or` keyword such as `userdn="ldap:///uid=admin,dc=example,dc=com` or `groupdn="ldap:///cn=administrators,ou=Groups,dc=example,dc=com`. The `not` keyword is also used to negate the result of a bind rule.

`userdn`

The `userdn` bind rule is used to target a requester based on their authenticated identity. The value of this bind rule must be in the form of an LDAP URL although in some cases it might not actually be a valid URL (for example, because the distinguished name (DN) element is not actually a valid DN). The LDAP URL can take several forms, including:

- It can have an LDAP URL that contains just a DN, without any wildcard, scope, or filter: "`ldap:///uid=admin,dc=example,dc=com`".
- It can contain an LDAP URL that contains a DN that has a pattern with asterisks as wildcards. Wildcards can be used as follows:
  - In place of an entire attribute value: `userdn="ldap:///uid=*,ou=People,dc=example,dc=com`
  - In place of a portion of an attribute value: `userdn="ldap:///uid=admin-*,ou=People,dc=example,dc=com`
  - In place of an attribute type: `userdn="ldap:///*=jdoe,ou=People,dc=example,dc=com`
  - In place of a single entire RDN component: `userdn="ldap:///uid=admin,*,dc=example,dc=com`
  - Two consecutive asterisks in place of any number of entire RDN components: `userdn="ldap:///uid=admin,**,dc=example,dc=com`
- It can have an LDAP URL that contains a parameterized DN. See the *Parameterized ACIs* section for more information about this.
- It can have an LDAP URL in which the DN is the string `anyone`: `userdn="ldap:///anyone`. This indicates that the ACI applies to any client, regardless of whether they are authenticated.
- It can have an LDAP URL in which the DN is the string `all`: `userdn="ldap:///all`. This indicates that the ACI applies to any authenticated client (regardless of the authentication identity), but not to unauthenticated or anonymous clients.
- It can have an LDAP URL in which the DN is the string `self`: `userdn="ldap:///self`. This indicates that the ACI applies to any operation in which the authenticated user is targeting their own entry.
- It can have an LDAP URL in which the DN is the string `parent`: `userdn="ldap:///parent`. This indicates that the ACI applies to any operation that targets an entry whose immediate parent is the requester's own entry. That is, it allows a requester to perform operations on entries immediately below their own.
- It can be a full LDAP URL, including a base DN (without wildcards or parameterization), scope, and filter suh as `userdn="ldap:///dc=example,dc=com??sub?(objectClass=person)"`.

The `userdn` bind rule can also contain multiple LDAP URLs. In that case, each URL should be separated by two consecutive vertical bar characters `||`: `("userdn="ldap:///uid=jdoe,ou=People,dc=example,dc=com || uid=jpublic,ou=People,dc=example,dc=com"")`.

The `!=` operator can be used as an alternative to the `=` operator: `userdn!="ldap:/// uid=admin,ou=People,dc=example,dc=com"`. In that case, the bind rule matches if the provided value does not match the authenticated identity.

`groupdn`

The `groupdn` bind rule is used to target a requester based on their group membership. The value of this bind rule must be an LDAP URL, such as `groupdn="ldap:///cn=Admin Users,ou=Groups,dc=example,dc=com"`. Only the DN element of the URL is used, and that DN must not contain any wildcards or parameterization.

The `groupdn` bind rule considers both direct group membership and nested membership such as if the user is a member of a group that is itself a member of a group listed in the `groupdn` value.

As with the `userdn` bind rule, the value of the `groupdn` bind rule can include multiple LDAP URLs separated by two consecutive vertical bar characters.

The `!=` operator can be used as an alternative to the `=` operator: `groupdn!="ldap:/// cn=Administrators,ou=Groups,dc=example,dc=com"`. In that case, the bind rule matches if authenticated user is not a member of the specified group.

`userattr`

The userattr bind rule can be used to target a requester based on their relation to the value of an attribute in the target entry. The value of the bind rule should contain the name of an attribute followed by the octothorpe character (#) and a keyword or attribute value.

There are four forms that userattr values can take:

- An attribute name followed by an octothorpe and the keyword `USERDN`, which indicates that the specified attribute should have a value that matches the DN of the authenticated user. For example, `userattr="manager#USERDN"` can be used to allow an operation if it targets an entry whose manager attribute has a value that matches the DN of the authenticated user.
- An attribute name followed by an octothorpe and the keyword `GROUPDN`, which indicates that the specified attribute should have a value that matches the DN of a group in which the authenticated user is a member. For example, `userattr="allowedEditors#GROUPDN"` can be used to allow an operation if it targets an entry whose `allowedEditors` attribute has a value that matches the DN of a group that contains the authenticated user either directly or indirectly through a nested group.
- An attribute name followed by an octothorpe and the keyword `LDAPURL`, in which case the value of the specified attribute must be an LDAP URL that is compared against the authenticated user's entry. For example, `userattr="allowedEditorCriteria#LDAPURL"` can be used to allow an operation if it targets an entry whose `allowedEditorCriteria` attribute has a value that is an LDAP URL whose base, scope, and filter matches the authenticated user's entry.
- An attribute name followed by an octothorpe and any value that is not one of `USERDN`, `GROUPDN`, or `LDAPURL`. In this case, that value is assumed to be an attribute value, and the operation can be authorized if both the authenticated user's entry and the target user's entry have that attribute value for the specified attribute. For example, `userattr="ou#Sales"` can be used to allow a user with an ou value of `Sales` to process operations against other users with an ou value of `Sales`.

When using the `USERDN` keyword, you can optionally specify a parent component to indicate that the target entry can be up to four levels below the authenticated user's entry. In this case, the word parent should be followed by an opening square bracket ([), a comma-delimited list of digits between 0 and 4 (inclusive), a closing square bracket (]), a period, and the rest of the value. A value of zero matches if the attribute value contains a DN that matches that of the authenticated user, while A value of four matches if the attribute value contains a DN that is exactly four levels below the authenticated user's entry. For example,

`userattr="parent[0,1,2,3,4].manager#USERDN"` can be used to allow an operation if it targets a user whose manager attribute contains a value that matches the DN of the authenticated user or is up to four levels below that DN.

The `!=` operator can be used as an alternative to the `=` operator such as `userattr!="manager#USERDN"`. In that case, the bind rule matches if the provided value does not match for the authenticated user.

`authmethod`

The `authmethod` bind rule is used to target a requester based on the mechanism that they used to authenticate to the server. The value for this element can take one of the following forms:

`none`

This matches if the requester is not authenticated; `authmethod="none"`.

`simple`

This matches if the requester authenticated using non-anonymous LDAP simple authentication; `authmethod="simple"`.

`ssl`

This matches if the requester authenticated using a client certificate; `authmethod="ssl"`. This is equivalent to `authmethod="SASL EXTERNAL"`.

`sasl` *<mechanisimName>*

This matches if the requester authenticated using the specified SASL mechanism: `authmethod="SASL PLAIN"`.

> ⓘ **Note:**
>
> The `ssl` value does not necessarily match just because the client is communicating with the server over a secure connection. Instead, it only matches if the client authenticated to the server with a client certificate chain using the SASL EXTERNAL mechanism.

The `!=` operator can be used as an alternative to the `=`operator: `authmethod!="simple"`. In that case, the bind rule matches if the client is not authenticated using the specified mechanism.

`ip`

The ip bind rule can be used to target a requester based on the IP address of the client system. The value can be a comma-delimited list of any of the following IP address patterns:

- A specific IPv4 address: `ip="1.2.3.4"`.
- An IPv4 address that uses the asterisk as a wildcard character to specify a range of addresses: `ip="1.2.3.*"`.
- An IPv4 address is followed by a plus sign and a subnet mask to specify a range of addresses: `ip="1.2.3.0+255.255.255.0"`.
- An IPv4 address using CIDR notation to specify a range of addresses: `ip="1.2.3.0/24"`.
- An IPv6 address as described in *RFC 2373*: `ip="0:0:0:0:0:0:0:1"`. IPv6 shorthand notation is also allowed: `ip="::1"`.

The `!=` operator can be used as an alternative to the “=” operator: `ip!="1.2.3.0/24"`. In that case, the bind rule matches if the client does not match the given IP address pattern.

`dns`

The `dns` bind rule can be used to target a requester based on a resolvable host name. The value can be a comma-delimited list of host names in either of the following forms:

- A complete resolvable host name: `dns="client.example.com"`.
- A host name that uses the asterisk as a wildcard in the leftmost component: `dns="*.example.com"`.

The `!=` operator can be used as an alternative to the `=` operator: `dns!="*.example.com"`. In that case, the bind rule matches if the client host name does not match the provided pattern.

In general, we do not recommend using the `dns` bind rule. DNS might be vulnerable to spoofing attacks, which could help an attacker gain unauthorized access to the system. Further, if name resolution is slow (for example, as a result of a network problem or an intentional denial of service attack), then that can adversely affect the performance of the PingDirectory Server.

`dayofweek`

The `dayofweek` bind rule is used to make access control decisions based on the current day of the week, as determined by the server's current time zone. The value should be a comma-delimited list containing one or more of the following values: `sun`, `mon`, `tue`, `wed`, `thu`, `fri`, `sat`. For example, `dayofweek="mon,tue,wed,thu,fri"`.

The `!=` operator can be used as an alternative to the `=` operator: `dayofweek!="sat,sun"`. In that case, the bind rule matches if the current day of the week is not included in the provided list.

`timeofday`

The `timeofday` bind rule is used to make access control decisions based on the current time of the day, as determined by the server's current time zone.

The value of this bind rule must be a four-digit number. The first two digits must represent the hour, with a value between `00` and `23`. The last two digits must represent the minute, with a value between `00` and `59`. For example, a value of `0123` represents a time of 1:23 a.m. in the server's time zone.

The operator for this bind rule can be one of the following:

=

The bind rule matches if the current time has the same hour and minute as specified in the value: `timeofday="0123"`.

!=

The bind rule matches if the current time has a different hour or minute than specified in the value: `timeofday!="0123"`.

<

The bind rule matches if the current time is between midnight (inclusive) and the time specified in the value (exclusive). That is, if the current time is earlier in the day than the specified time: `timeofday<"0123"`.

<=

The bind rule matches if the current time is between midnight (inclusive) and the time specified in the value (inclusive). That is, if the current time is earlier in the day or the same as the specified time: `timeofday<="0123"`.

>

The bind rule matches if the current time is between the time specified in the value (exclusive) and 11:59 p.m. (inclusive). That is, if the current time is later in the day than the specified time: `timeofday>"0123"`.

>=

The bind rule matches if the current time is between the time specified in the value (inclusive) and 11:59 p.m. (inclusive). That is, if the current time is later in the day or the same as the specified time: `timeofday>="0123"`.

`oauthscope`

The oauthscope bind rule is used to make access control decisions based on the set of scopes associated with an OAuth 2.0 access token that the client used to authenticate.

This bind rule is primarily used for clients that have authenticated with the OAUTHBEARER SASL mechanism or for clients that have authenticated to Delegated Admin using an OAuth token. It does not currently apply to clients using SCIM or the Directory REST API even if they have authenticated with an OAuth token.

The value for this bind rule might have one of the following forms:

- It can be the name of a single scope to match: `oauthscope="admin_user"`.
- It can be a substring assertion that contains one or more asterisks to use as wildcards: `oauthscope="admin_*"`.
- It can be a single asterisk to indicate that any scope is sufficient: `oauthscope="*"`.

The `!=` operator can be used as an alternative to the `=` operator: `oauthscope!="admin_user"`. In that case, the bind rule matches if the user does not have the specified scope.

Parameterized ACIs

Parameterized ACIs are useful for cases in which the data in a PingDirectory Server instance has the same structure repeated many times, and when each structure needs to have a similar set of access control rules.

This is especially common in a multi-tenant environment in which users within a tenant might need access to other entries within the same tenant, but not to other entries outside their organization.

For example, consider a server that has a DIT structure like the following:

- `dc=example, dc=com`
  - `ou=tenants`
    - `ou=Company A`
      - `ou=People`
      - `ou=Groups`
        - `cn=Administrators`
    - `ou=Company B`
      - `ou=People`
      - `ou=Groups`
        - `cn=Administrators`
    - `ou=Company C`
      - `ou=People`
      - `ou=Groups`
        - `cn=Administrators`

In each case, members of the `cn=Administrators,ou=Groups,ou=<companyName>,ou=tenants,dc=example,dc=com` group might need to be able to manage entries after `ou=<companyName>,ou=tenants,dc=example,dc=com`. While it might be possible to accomplish

this by creating similar ACIs throughout the DIT (one for each tenant), this can also be accomplished by creating one parameterized ACI like the following example.

```
(target="ldap:///ou=($1),ou=tenants,dc=example,dc=com")
(version 3.0; acl "Allow organization administrators to manage
 entries in their organization"; allow (all) groupdn="ldap:///
cn=Administrators,ou=Groups,ou=($1),ou=tenants,dc=example,dc=com";)
```

In this case, the "($1)" is a placeholder that matches between the `target` and `groupdn` elements of the access control rule. If the client is authenticated as a user who is a member of any group that matches that pattern in the `target` bind rule, then the value that matches the placeholder within that pattern is also substituted in place of the same pattern within the target element.

Parameterized ACIs can also be used in conjunction with the `userdn` bind rule. For example, the following ACI grants any user within the organization permission to access a select set of attributes from any user within the same organization.

```
(target="ldap:///ou=($1),ou=tenants,dc=example,dc=com")(targetattr="uid||
cn||givenName||sn||mail")(version 3.0; acl "Allow users within an
 organization to access select attributes from other entries in the
 same organization"; allow (read,search,compare) userdn="ldap:///
uid=($2),ou=People,ou=($1),ou=tenants,dc=example,dc=com";)
```

Parameterized DNs used in the `userdn` or `groupdn` bind rules can have multiple placeholders. Not all of those placeholders need to be used in the target.

Defining ACIs in user data

Most access control rules should be defined in the user data.

This is especially true for ACIs that apply to user data. This is accomplished by defining the rules in the ACI operational attribute in entries within the DIT.

For example, the following LDIF modify change record can be used to add an access control rule to the `ou=People,dc=example,dc=com` entry that allows users within that branch to update their own password.

```
dn: ou=People,dc=example,dc=com
changetype: modify
add: aci
aci: (targetattr="userPassword")(version 3.0; acl "Allow users to update
  their own password"; allow (write) userdn="ldap:///self";)
```

Ideally, ACIs should be placed in the entry that is the base of the subtree to which it applies. For example, if an ACI applies to entries at or below `ou=People,dc=example,dc=com`, then it should be defined in the ACI attribute in the `ou=People,dc=example,dc=com` entry. While such an ACI could be defined at a higher level in the DIT, like `dc=example,dc=com`, and use the target element to indicate that it applies to a specified subtree, we generally recommend keeping ACIs as close to the data that they manage as possible.

ACIs cannot be placed in a different subtree than the data to which they apply. For example, it is not possible to define an ACI in the `ou=People,dc=example,dc=com` entry if it is intended to apply to entries in the `ou=Groups,dc=example,dc=com` branch.

Defining global ACIs

Access control rules can also be defined in the server configuration and in particular in the `global-aci` property of the access control handler configuration.

This should generally be limited to access control rules that meet one or more of the following criteria:

- They need to apply to entries in backends other than those containing user data. This includes the root DSE, the server configuration, monitor entries, the LDAP changelog, administrative tasks, and other areas of the server. If these ACIs apply to data in a specific backend, then the `target` keyword should be used to limit the scope of the rule.
- They need to apply to one or more extended operations (using the `extop` target). ACIs that grant or deny access to extended operations must be defined in the global configuration.
- They need to apply to request controls (using the `targetcontrol` target). Although it may be possible to define ACIs pertaining to request controls in user data (especially if those controls are only expected to be used when issuing requests targeting user data), ACIs pertaining to request controls are commonly placed in the global configuration.

For example, the following configuration change can be used to define a global ACI that grants members of the “Changelog Readers” group permission to read entries in the LDAP changelog.

```
dsconfig set-access-control-handler-prop --add 'global-aci:
(target="ldap:///cn=changelog")(version 3.0; acl "Allow changelog read
 access"; allow (read,search,compare) groupdn="ldap:///ou=Changelog
 Readers,ou=Groups,dc=example,dc=com";)'
```

The get effective rights request control

After you have defined your access control policy, we recommend that you verify that it is working as expected.

While you can do this by issuing requests against the server to ensure that operations are permitted and rejected as appropriate, the PingDirectory Server also provides support for a get effective rights request control that can be used to determine what access a given user has to a specified entry.

This control can be used programmatically through the *UnboundID LDAP SDK for Java*, but it can also be done from the command line using the ldapsearch tool. The tool provides the following arguments pertaining this feature:

`--getEffectiveRightsAuthzID`

Identifies the user whose access control rights should be examined. This should be an authorization ID that either identifies the user by distinguished name (DN) (prefixed by `dn:`) or username (prefixed by `u:`).

`--getEffectiveRightsAttribute`

Specifies the name of an attribute for which you wish to obtain the specified user’s effective rights. This argument can be used multiple times to provide multiple attribute names.

For example:

```
$ bin/ldapsearch --hostname ds.example.com \
     --port 636 \
     --useSSL \
     --bindDN "cn=Directory Manager" \
     --baseDN dc=example,dc=com \
     --scope base \
     --getEffectiveRightsAuthzID
 dn:uid=test.user,ou=People,dc=example,dc=com \
     --getEffectiveRightsAttribute objectClass \
     --getEffectiveRightsAttribute dc \
     "(objectClass=*)" \
     aclRights
Enter the bind password:

dn: dc=example,dc=com
aclRights;attributeLevel;objectclass:search:1,read:1,compare:1,write:0,
 selfwrite_add:0,selfwrite_delete:0,proxy:0
```

```
aclRights;attributeLevel;dc: search:1,read:1,compare:1,write:0,
 selfwrite_add:0,selfwrite_delete:0,proxy:0
aclRights;entryLevel: add:0,delete:0,read:1,write:0,proxy:0

# Result Code:  0 (success)
# Number of Entries Returned:  1
```

Each search result entry that is returned includes an `aclRights` attribute that indicates what rights the target user has when interacting with that entry. If you do not use the `--getEffectiveRightsAttribute` argument to specify any attribute names, then only the `aclRights;entryLevel` attribute is used to show the rights the user has when interacting with the entry itself will be returned. Otherwise, there is an additional `aclRights;attributeLevel` value for each requested attribute showing the rights for that attribute.

Debugging ACI issues

If you encounter a scenario in which the server seems to exhibit a behavior that is not in-line with the expected access control configuration, you can use the Debug ACI Logger to obtain detailed information about the access control decisions the server is making.

You can do this with the following configuration change.

```
dsconfig set-log-publisher-prop \
     --publisher-name "Debug ACI Logger" \
     --set enabled:true
```

After you enable it, this logger records information about its access control decisions to the `logs/debug-aci` file. Because the server can write huge amounts of data to this file on a busy production server, you might want to try this on a separate instance that has been populated with the same set of access control rules. Alternatively, you can configure the logger with criteria that only matches the operations you are trying to investigate.

Other ways of restricting requests and data access

To preserve security and privacy, applications should only be permitted to perform the bare minimum set of operations that they need, and they should only be permitted to access a minimal set of information in the entries that they are allowed to access.

It is also critical to severely restrict what unauthorized users are allowed to do. PingDirectory Server provides several mechanisms to help with this:

- It has a fine-grained access control subsystem that are used to indicate which requests are allowed and which data can be retrieved from the server.
- It has a privilege subsystem that can be used to require additional authorization when processing certain types of operations or to grant additional capabilities to some users.
- Client connection policies can be used to impose limits on the types of operations that clients are allowed to request, even restricting what is possible for root users and topology administrators.
- Sensitive attributes can also impose strong restrictions on access to certain attribute types and can also impose restrictions on root users and topology administrators.

Rejecting unauthenticated requests

One of the best ways to prevent unauthorized access to the server is to require authentication for all operations processed in the server.

If the server is configured to reject unauthenticated requests, then attackers would either need to legitimate access to an account in the server, or they would need to somehow obtain credentials for a valid account.

The global configuration makes it easy to reject requests from unauthenticated clients through the following properties:

reject-unauthenticated-requests

Indicates whether the server rejects requests from unauthenticated clients, including clients that have not yet authenticated, clients whose most recent authentication attempt failed, or clients whose most recent authentication attempt was an anonymous bind.

allowed-unauthenticated-requests-criteria

Specifies an optional set of criteria used to indicate that certain operations are allowed over an unauthenticated connection.

Even if `reject-unauthenticated-requests` is true, then the server allows a small number of requests from unauthenticated connections. These include:

- Bind requests, which are used to authenticate connections.
- StartTLS extended requests, which are used to add TLS encryption to initially insecure connections.
- The start administrative session extended request, which is used to indicate that subsequent operations are part of an administrative session. When it is used, this should be the first operation on the connection, even before bind and StartTLS operations. There is no inherent security risk in allowing this for unauthenticated clients.

If any other types of requests should be allowed for unauthenticated clients, then the `allowed-unauthenticated-requests-criteria` property should be used to define criteria that matches only those operations.

Privileges

PingDirectory Server defines a number of privileges that it can use to give a user additional functionality or restrict access to some functionality.

Available privileges

Some of the defined privileges include in the following.

Privilege	Description
`audit-data-security`	Required for a user to invoke the audit data security task to generate a report on security-related aspects of the data contained in the server.
`backend-backup`	Required to initiate an online backup through an administrative task.
`backend-restore`	Required to initiate an online restore through an administrative task.
`bypass-acl`	Exempts the user from access control evaluation for all operations. This grants the user full access to all data in the server, although they might still be limited by things like client connection policies or sensitive attributes.

Privilege	Description
`bypass-pw-policy`	Exempts the user from certain password policy restrictions when changing another user's password. This includes: ▪ The user is allowed to set a pre-encoded password for another user even if the password policy forbids it. ▪ The user is allowed to set a password for another user even if it fails validation. ▪ The user is allowed to set a password for another user even if it is in the user's password history.
`bypass-read-acl`	Exempts the user from access control evaluation for read operations, including search and compare. Write operations are still subject to access control evaluation, and the user might still be limited by constraints in the client connection policy and sensitive attribute definitions.
`collect-support-data`	Required to invoke the `collect-support-data` tool through an administrative task or an extended operation.
`config-read`	Required for a user to to read any information from the server configuration.
`config-write`	Required (in addition to the `config-read` privilege) to update the server configuration.
`disconnect-client`	Required to forcefully disconnect another client.
`exec-task`	Required to invoke an exec task.
`file-servlet-access`	Might be required to access the content of certain file servlet instances, including the instance root file servlet.
`jmx-notify`	Required to subscribe to receive JMX notifications.
`jmx-read`	Required to read monitor data from JMX.
`ldif-export`	Required to initiate an online LDIF export through an administrative task.
`ldif-import`	Required to initiate an online LDIF import through an administrative task.
`lockdown-mode`	Required to cause the server to enter and leave lockdown mode, and also to submit requests while the server is in lockdown mode.

Privilege	Description
`manage-topology`	Required to process topology-related operations, like adding servers to and removing servers from the topology.
`modify-acl`	Required to add and remove ACIs.
`password-reset`	Required to change the password for another user. This privilege is also required to use the password policy state extended operation and might be required for other password-policy-related operations. Either this privilege or the `permit-externally-processed-authentication` privilege is required to use the UNBOUNDID-EXTERNALLY-PROCESSED-AUTHENTICATION SASLmechanism.
`permit-externally-processed-authentication`	Either this privilege or the password-reset privilege is required to be able to use the UNBOUNDID-EXTERNALLY-PROCESSED-AUTHENTICATION SASL mechanism.
`permit-get-password-policy-state-issues`	Required to use the get password policy state issues request control.
`privilege-change`	Required to alter the set of privileges assigned to a user.
`proxied-auth`	Required to request an alternate authorization identity (that is, to impersonate another user). This includes the ability to use the proxied authorization request control, the intermediate client request control with a userIdentity value, and requesting an alternate authorization identity in applicable SASL mechanisms.
`server-restart`	Required to initiate an online restart through an administrative task.
`server-shutdown`	Required to initiate a server shutdown through an administrative task.
`soft-delete-read`	Required to access soft-deleted entries.
`stream-values`	Required to use the stream directory values or stream proxy values extended operation.
`third-party-task`	Required to invoke a custom task implemented using the Server SDK.
`unindexed-search`	Required to request an unindexed search.

Privilege	Description
`unindexed-search-with-control`	Required to request an unindexed search in conjunction with the permit unindexed search request control.
`update-schema`	Required to update the server schema.
`use-admin-session`	Required to create an administrative session that allows operations to be processed in a dedicated thread pool.

Assigning privileges

Privileges can be assigned to users by adding the `ds-privilege-name` operational attribute to a user's entry with a value set to the desired privilege. This is a multivalued attribute, so multiple privileges can be assigned.

For example, the following modification demonstrates the process for granting the password-reset privilege to a user. The `privilege-change` privilege is required to alter the set of privileges assigned to a user, so this modification is only allowed if the requester has that privilege.

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

This process also works for root users and topology administrators although you can also use `dsconfig` or the admin console to alter the set of privileges for those users through the `privilege` property in the user configuration.

Root users and topology administrators can also automatically inherit a default set of privileges from the configuration. This default set of privileges is defined in the `default-root-privilege-name` property of the Root DN configuration object. If a root user or topology administrator is to automatically inherit this default set of privileges, then their configuration object has the `inherit-default-root-privileges` property set to true.

Client connection policy restrictions

Client connection policies can be used to set hard limits on what types of requests clients are allowed to issue.

This applies to all clients associated with the policy, regardless of what privileges they have and what access control rights they have been granted. If a client connection policy prohibits something, then not even root users or topology administrators are permitted to do it if they are using a connection associated with that policy.

The following client connection properties can be used to restrict the set of requests that clients can issue.

Property	Description
`allowed-operation`	The types of operations that clients associated with the policy are allowed to request. Allowed values include any non-empty combination of the following: `abandon`, `add`, `bind`, `compare`, `delete`, `extended`, `modify`, `modify-dn`, and `search`. By default, all operation types are allowed.

Property	Description
`required-operation-request-criteria`	A set of criteria that requests must match in order to be processed by the server. If required operation request criteria is defined and a client associated with the policy issues a request that does not match that criteria, then the operation is rejected. By default, no required operation request criteria are defined.
`prohibited-operation-request-criteria`	A set of criteria that requests must not match in order to be processed by the server. If a prohibited operation request criteria is defined and a client associated with the policy issues a request matching that criteria, then the operation is rejected. By default, no prohibited operation request criteria are defined.
`prohibited-operation-request-criteria`	A set of criteria that requests must not match in order to be processed by the server. If a prohibited operation request criteria is defined and a client associated with the policy issues a request matching that criteria, then the operation is rejected. By default, no prohibited operation request criteria are defined.
`allowed-request-control`	The OIDs of the controls that clients associated with the policy will be allowed to include in requests. If any allowed request control OIDs are defined, then any request that contains a control other than one of those listed is rejected. By default, no allowed request control OIDs are defined, which indicates that any control not included in the set of denied request controls is permitted.
`denied-request-control`	The OIDs of the controls that clients associated with the policy are not allowed to include in requests. If any denied request control OIDs are defined, then any request that contains a control whose OID is contained in this set IS rejected. By default, no denied request control OIDs are defined.
`allowed-extended-operation`	The OIDs of the extended operations that clients are allowed to request. If any allowed extended operation OIDs are defined, then any extended request that uses an OID other than one of those listed is rejected. By default, no allowed extended operation OIDs are defined, which indicates that any request not included in the set of denied extended operations is permitted.

Property	Description
`denied-extended-operation`	The OIDs of the extended operations that clients are not allowed to request. If any denied extended operation OIDs are defined, then any request that contains a control whose OID is included in this set is rejected. By default, no denied extended operation OIDs are defined.
`allowed-auth-type`	The types of authentication that clients are allowed to request. Allowed values include `simple` and `sasl`, and both are allowed by default.
`allowed-sasl-mechanism`	The names of the SASL mechanisms that clients are allowed to use to authenticate. If any allowed SASL mechanism names are defined, then any SASL bind attempt that uses a mechanism not included in this list is rejected. By default, no allowed SASL mechanism names are defined, which indicates that any SASL mechanism not included in the set of denied mechanisms is permitted.
`denied-sasl-mechanism`	The names of the SASL mechanisms that clients are not allowed to use to authenticate. If any denied SASL mechanism names are defined, then any SASL bind attempt that uses one of those mechanisms is rejected. By default, no denied SASL mechanism names are defined.
`allowed-filter-type`	The types of search filters that clients are allowed to use for searches with a scope other than baseObject (searches with a baseObject scope are allowed to use any kind of filter, as those searches are always be efficient to process). Allowed values include any non-empty combination of the following: `and`, `or`, `not`, `compare`, `equality`, `sub-initial`, `sub-any`, `sub-final`, `greater-or-equal`, `less-or-equal`, `present`, `approximate-match`, and `extensible-match`. By default, all filter types are allowed.
`minimum-substring-length`	The minimum number of consecutive non-wildcard bytes that must be present in each subInitial, subAny, and subFinal element of a substring filter component. Any attempt to use a substring filter with an element containing fewer than this number of non-wildcard bytes is rejected. By default, no minimum substring length is enforced.

Property	Description
`allow-unindexed-searches`	Indicates whether clients associated with the policy are allowed to request unindexed searches. If this is set to true (which is the default), then unindexed search operations are permitted in cases in which at least one of the following is true: ▪ The requester has the `unindexed-search` privilege. ▪ The `unindexed-search` privilege is disabled in the global configuration. ▪ The requester has the `unindexed-search-with-control` privilege and the request includes the permit unindexed search request control. ▪ The `unindexed-search-with-control` privilege is disabled in the global configuration and the request includes the permit unindexed search request control.
`allow-unindexed-searches-with-control`	Indicates whether clients associated with the policy are allowed to request unindexed searches as long as the request also includes the permit unindexed search request control. If this is set to true (which is the default), then unindexed search operations are permitted in cases in which the requester has either the `unindexed-search` privilege or the `unindexed-search-with-control` privilege (or at least one of those privileges is disabled in the global configuration) and the request includes the permit unindexed search request control. ⓘ **Note:** If this property is set to true, then unindexed searches can be allowed for authorized requests that include the permit unindexed search request control even if the `allow-unindexed-searches` property is set to false.

Sensitive attributes

PingDirectory Servers allow you to create sensitive attribute definitions, which provide enhanced protection for interactions with a specified set of attributes.

You can prevent the values of sensitive attributes from being retrieved at all, or you can ensure that they are only retrieved over secure connections, and similar protections are available for other kinds of read and write operations. Sensitive attribute definitions can be imposed for all clients, even for root users and topology administrators.

Sensitive attribute definitions include the following configuration properties.

Property	Description
`attribute-type`	The names or OIDs of the attribute types that are to be governed by this sensitive attribute definition. If the same attribute type is included in multiple sensitive attribute definitions, it is subject to the most strict intersection of those definitions.
`include-default-sensitive-operational-attributes`	Indicates whether the server should automatically declare a predefined set of special operational attributes as sensitive under the conditions of this sensitive attribute definition. At present, this includes the ds-sync-hist attribute, which is used to hold historical information for replication conflict resolution, and it can include former values for some attributes in the entry. This is true by default.
`allow-in-returned-entries`	Indicates whether to allow the specified attributes to be included in search result entries that are returned to clients (or other forms of entries that can be returned, like the “before” and “after” versions of an entry when used with the pre-read and post-read request controls). Allowed values include: ▪ `secure-only` — Only allow the attributes to be returned to clients over a secure connection (for example, one encrypted with TLS). The attributes are stripped from search result entries over an insecure connection. ▪ `suppress` — Do not allow the attributes to be returned to clients, even over a secure connection. The attributes are always stripped from search result entries. ▪ `allow` — Allow the attributes to be returned to clients over a secure or insecure connection.
`allow-in-filter`	Indicates whether to allow the specified attributes to be included in search filters. If an attribute can be used in a filter, then clients with the ability to read the entry might be able to determine its values through brute force, by simply trying different values until a match is found. They can also be used to find all entries with a given value. Allowed values include: ▪ `secure-only` — Only allow the attributes to be used in filters for requests sent over a secure connection. Search requests that use a sensitive attribute in the filter are rejected if they are received over an insecure connection. ▪ `reject` — Reject any search request that includes the sensitive attribute in a filter, even over a secure connection. ▪ `allow` — Allow the attributes to be used in filters over a secure or insecure connection.

Property	Description
`allow-in-add`	Indicates whether to allow the specified attributes to be included in add requests. Allowed values include `secure-only`, `reject`, and `allow`.
`allow-in-compare`	Indicates whether to allow the specified attributes to be used in compare assertions. As with search filters, compare assertions can be used to try to discover attribute values through brute force. Allowed values include `secure-only`, `reject`, and `allow`.
`allow-in-modify`	Indicates whether to allow the specified attributes to be updated in modify requests. Allowed values include `secure-only`, `reject`, and `allow`.

By default, the server includes sensitive attribute definitions that prevent clients from accessing passwords and other sensitive authentication information like generated one-time passwords and TOTP shared secrets. They prevent clients from retrieving attribute values or using them in search filters or compare assertions, even over a secure connection, and they only allow the values to be included in add and modify requests that are received over a secure connection. However, these sensitive attribute definitions are not enforced by default.

To ensure that the server enforces the restrictions imposed by sensitive attribute definitions, they must be associated with one or more client connection policies. The following property in the global configuration can enable specified sensitive attribute definitions across all client connection policies:

`sensitive-attribute`

The set of sensitive attribute definitions that should be enabled by default across all client connection policies.

You can also customize the set of client connection policies in which sensitive attribute definitions should be enforced. This is done through the following properties in the client connection policy configuration:

`sensitive-attribute`

The set of sensitive attribute definitions that should be enabled for connections using that client connection policy.

`exclude-global-sensitive-attribute`

The set of sensitive attribute definitions that are enabled in the global configuration that should not be enforced for clients using that client connection policy.

We recommend preventing clients (even root users or topology administrators) from being able to retrieve passwords or other kinds of authentication secrets, and to only allow them to be updated over secure connections. This can be done by updating the global configuration to enable those sensitive attribute definitions across all client connection policies using a change, as in the following example.

```
dsconfig set-global-configuration-prop \
     --add "sensitive-attribute:Sensitive Password Attributes" \
     --add "sensitive-attribute:Delivered One-Time Password" \
     --add "sensitive-attribute:TOTP Shared Secret"
```

However, you might have an application that has a legitimate need to access encoded passwords. For example, you might want to use the PingDataSync Server to synchronize passwords from PingDirectory to other data stores. In such cases, the best solution might be to create a custom client connection policy

with connection criteria that only matches that application, and then configure it to exclude the appropriate sensitive attribute from the global configuration, as in the following example.

```
dsconfig set-client-connection-policy-prop \
     --policy-name "Synchronization Application" \
     --add "exclude-global-sensitive-attribute:Sensitive Password
 Attributes"
```

See the `config/use-sensitive-attributes-to-prevent-password-access.dsconfig` sample batch file for more information.

Writability mode

Use writability mode to indicate whether the server allows clients to update the data in the server.

This can be configured through the `writability-mode` property in the global configuration, or through the `writability-mode` property for each backend. In either case, the property offers the following values:

`enabled`

Indicates that writes are enabled.

`disabled`

Indicates that all write attempts are rejected, regardless of their origin.

`internal-only`

Indicates that write attempts from external clients are rejected, but writes received from replication or initiated internally within the server (for example, as a result of password policy state processing).

ⓘ **Note:**

The writability mode defined in the global configuration applies only to user data backends. It will not apply to private backends, like the server configuration or schema.

If the `writability-mode` values differ between the global configuration and the backend configuration, then the server uses whichever property is more restrictive. That is, if either one is set to disabled, then all write attempts in that backend are rejected. If one is enabled and the other is set to `internal-only`, then the `internal-only` mode is used for that backend.

If you want to configure a PingDirectory Server instance to be a read-only replica, then you should use the `internal-only` writability mode so that replication changes are still accepted but writes from external clients are rejected. If you would prefer that the server generate a referral for external each write attempt rather than rejecting it outright, then you should create an instance of the “Referral on Update” plugin and specify the base referral URLs that the server should use.

User resource limits

PingDirectory Server can impose limits on client connections to restrict the amount of data they can retrieve or limit their ability to request expensive operations.

This includes:

- The maximum number of entries they can retrieve in any search operation
- The maximum number of entries the server considers when processing any search operation
- The maximum length of time the server is allowed to spend while processing a search operation
- The maximum length of time that a client connection can remain idle between requests

- The maximum number of entries that can be joined with any single search result entry when performing a search using the LDAP join request control

Default values for each of these limits can be defined in the global configuration, but they can be overridden on a per-user basis with operational attributes in the user's entry. Hard upper bounds for these limits for these properties may also be imposed by client connection policies, and the client can set upper bounds for the size limit and time limit when submitting a search request.

Defining resource limits in the global configuration

Use the following global configuration properties to enforce resource limits and provide protection against denial-of-service attacks.

Property	Description
`size-limit`	The default maximum number of entries that a client can retrieve in any single search operation. If the client attempts to process a search that matches more than 1000 entries, then the operation fails with a `sizeLimitExceeded` result (after returning all matching entries identified before the size limit was reached). This is set to 1,000 by default. A value of zero indicates that no limit should be enforced.
`time-limit`	The default maximum length of time in seconds that the server is allowed to spend processing any single search operation. If the client requests a search operation that takes longer than this length of time to complete, then the operation fails with a `timeLimitExceeded` result (after returning any matching entries identified before the time limit was reached). This is set to one minute by default. A value of zero seconds indicates that no limit should be enforced.
`idle-time-limit`	The default maximum length of time that a client connection should be allowed to remain established after the server has completed processing on the most recent request issued on that connection. If the client remains idle for longer than this duration, the connection is terminated. A value of zero seconds (which is the default value) indicates that no idle time limit should be enforced.

Property	Description
`lookthrough-limit`	The default maximum number of entries that the server can examine in the course of processing a search request, regardless of whether those entries actually match the search criteria. This is primarily useful for limiting resource consumption when processing unindexed searches, and if the client attempts to process a search operation in which the server needs to examine more than this number of entries, then the operation fails with an `adminLimitExceeded` result (after returning any matching entries identified before the lookthrough limit was reached). This is set to 5,000 by default. A value of zero indicates that no limit should be enforced.
`ldap-join-size-limit`	The default maximum number of entries that can be joined with each search result entry when processing a search using the LDAP join request control. If an entry is joined with more than this number of entries, then the join result control for that entry has a sizeLimitExceeded result code (and might or might not include any matching entries). This is set to 10,000 by default. A value of zero indicates that no limit should be enforced.
`maximum-concurrent-connections`	The maximum number of connections that can be established to the server at any one time. Once this limit is reached, then any subsequent connection attempts are rejected until an existing client connection has been closed. A value of zero (which is the default) indicates that no limit is imposed by the server, although the operating system can impose a limit (for example, based on the number of available file descriptors).
`maximum-concurrent-connections-per-ip-address`	The maximum number of connections that can be established to the server at any one time from the same client IP address. Once this limit has been reached, then any subsequent attempts to establish a connection from that client are rejected until an existing connection from that client has been closed. A value of zero (which is the default) indicates that no limit is enforced.
`maximum-concurrent-connections-per-bind-dn`	The maximum number of connections that can be established to the server at any one time while authenticated as the same user. Once this limit has been reached, then any subsequent attempts to bind as that user cause the connection to be terminated until an existing connection authenticated as that user is closed or binds as a different user. A value of zero (which is the default) indicates that no limit is enforced.

Property	Description
`maximum-concurrent-unindexed-searches`	The maximum number of unindexed searches that can be in progress within the server at any one time. If an unindexed search is requested while the maximum number of searches are already in progress, then that search is rejected with an `unwillingToPerform` result. This is set to ten by default. A value of zero indicates that no limit is enforced.

You can also configure the server to suppress duplicate error log messages and administrative alerts. This can help prevent flooding the log with repeated messages if the same condition keeps occurring. The global configuration properties used to control this are:

Property	Description
`duplicate-error-log-limit` and `duplicate-error-log-time-limit`	The maximum number of error log messages of the same type that can be written within a given length of time. If that error log message rate is exceeded, then any additional error log messages of that type within that time period are suppressed, and the server writes a message at the end of that time indicating the number of messages that were suppressed. By default, the server starts suppressing error log messages after 2,000 messages of the same type in a five-minute period.
`duplicate-alert-limit` and `duplicate-alert-time-limit`	The maximum number of administrative alerts of the same type that can be generated within a given length of time. If that alert rate is exceeded, then any additional alerts of that type within that time period are suppressed, and the server generates an additional alert at the end of that time period indicating the number of alerts that were suppressed. By default, the server starts suppressing alerts after 10 alerts of the same type are generated in a ten-minute period.

Defining resource limits in operational attributes

Although the global configuration defines default values for several resource limits, it is possible to override those default values on a per-user basis by adding an appropriate set of operational attributes to the user's entry.

Resource limits set through operational attributes can grant a user a higher limit than is available by default in the global configuration, or it can impose a lower limit than would otherwise be permitted by default.

These operational attributes include:

Attribute	Description
`ds-rlim-size-limit`	The maximum number of entries that the user is allowed to retrieve in any single search operation. A value of zero indicates that no size limit is enforced for the user. If this attribute is present in a user's entry, then it overrides the default `size-limit` value from the global configuration.
`ds-rlim-time-limit`	The maximum length of time in seconds that the server is allowed to spend processing any single search operation for the user. A value of zero indicates that no time limit is enforced for the user. If this attribute is present in a user's entry, then it overrides the default `time-limit` value from the global configuration.
`ds-rlim-lookthrough-limit`	The maximum number of entries that the server is allowed to examine while processing any single search operation for the user. A value of zero indicates that no lookthrough limit is enforced for the user. If this attribute is present in a user's entry, then it overrides the default `lookthrough-limit` value from the global configuration.
`ds-rlim-idle-time-limit`	The maximum length of time in seconds that the user is allowed to have an idle connection (one in which no operations in progress) established. A value of zero indicates that no idle time limit is enforced for the user. If this attribute is present in a user's entry, then it overrides the default idle-time-limit value from the global configuration.
`ds-rlim-ldap-join-size-limit`	The maximum number of entries that are joined with any single search result entry when processing a search request that includes the LDAP join request control. A value of zero indicates that no LDAP join size limit is enforced for the user. If this attribute is present in the user's entry, then it overrides the default `ldap-join-size-limit` from the global configuration.

Each of these attributes can be explicitly set in the entries for users that should have a value that is different from the corresponding property in the global configuration. You can also use virtual attributes to dynamically assign values for these attributes using criteria like the location or content of the user's entry or the groups in which that user is a member or the client connection policy to which they are assigned.

Defining resource limits in client connection policies

Use client connection policies to impose hard upper bounds on the values of some resource limit properties.

If the client connection policy is configured with a resource limit that is lower than the limit that would otherwise be imposed for the associated client, then the client connection policy's lower limit is enforced (even for root accounts and other types of administrative accounts). However, the client connection policy never grants a client a higher limit than it would otherwise have.

The client connection policy properties that are used to define resource limits include the following.

Property	Description
`maximum-concurrent-connections`	The maximum number of client connections that can be associated with the client connection policy at any one time. If the maximum number of connections are already associated with the policy, any attempt to assign another connection to the policy causes that connection to be terminated. A value of zero (which is the default) indicates that no limit is enforced.
`maximum-connection-duration`	The maximum length of time that connections associated with the policy can be established, regardless of how active those connections are. Any connection that is established for longer than this period of time will be terminated. A value of zero seconds (which is the default) indicates that no maximum connection duration is enforced.
`maximum-idle-connection-duration`	The maximum length of time that connections associated with the policy are allowed to remain idle (without issuing any new requests). Any client connection that is idle for longer than this period of time is terminated. A value of zero seconds (which is the default) indicates that no maximum idle connection duration is enforced.
`maximum-operation-count-per-connection`	The maximum number of requests that connections associated with the policy are allowed to request over the life of that connection. Once a connection has already requested the maximum number of operations, if it attempts to request any other operations, then that connection is terminated. A value of zero (which is the default) indicates that no maximum operation count is enforced.
`maximum-concurrent-operations-per-connection`	The maximum number of operations that a client associated with the policy can request at any one time. Once the maximum number of concurrent operations are already active for a connection, then any new requests can optionally block for a period of time (specified by the `maximum-concurrent-operation-wait-time-before-rejecting` property) to see if any of the outstanding operations complete. At that point, the request can be rejected or the connection can be terminated based on the value of the `maximum-concurrent-operations-per-connection-exceeded-behavior` property. By default, no maximum concurrent operation limit is imposed.

Property	Description
`maximum-connection-operation-rate`	The maximum rate at which a client can issue requests. If provided, then the value should be provided as a count followed by a slash and a time duration (for example, 100/s indicates a maximum rate of one hundred requests per second, while 10K/6h indicates a maximum rate of 10,000 requests over a six-hour period). If any connection exceeds this rate, subsequent requests within that time period can be rejected or the connection can be terminated, as controlled by the `connection-operation-rate-exceeded-behavior` property. By default, no maximum connection operation rate is enforced.
`maximum-policy-operation-rate`	The maximum rate at which all clients associated with the client connection policy can issue requests. If provided, then the value should be provided as a count followed by a slash and a time duration. If the maximum policy operation rate is exceeded, then subsequent requests within that time period can be rejected or the connection can be terminated, as controlled by the `policy-operation-rate-exceeded-behavior` property. By default, no maximum connection operation rate is enforced.
`maximum-search-size-limit`	The maximum number of entries that can be returned in response to any single search operation for clients associated with the client connection policy. A value of zero (which is the default) indicates that the policy does not impose a maximum size limit for client connections, and they are subject to whatever limit is in place through the global configuration or operational attributes in the authenticated user's entry.
`maximum-search-time-limit`	The maximum length of time that the server can spend processing any single search operation for clients associated with the client connection policy. A value of zero seconds (which is the default) indicates that the policy does not impose a maximum time limit for client connections, and they are subject to whatever limit is in place through the global configuration or operational attributes in the authenticated user's entry.

Property	Description
`maximum-search-lookthrough-limit`	The maximum number of entries that the server can examine when processing any single search operation for clients associated with the client connection policy. A value of zero (which is the default) indicates that the policy does not impose a maximum lookthrough limit for client connections, and they aer subject to whatever limit is in place through the global configuration or operational attributes in the authenticated user's entry.
`maximum-ldap-join-size-limit`	The maximum LDAP join size limit that is enforced for clients associated with the client connection policy. A value of zero (which is the default) indicates that the policy does not impose a maximum join size limit for client connections, and they are subject to whatever limit is in place through the global configuration or operational attributes in the authenticated user's entry.
`maximum-sort-size-limit-without-vlv-index`	The maximum number of entries that the server attempts to sort without the benefit of a VLV index. If the client issues a search request that includes the server-side sort control and matches more than this number of entries, then the server either returns the results in unsorted form (if the sort request control is not marked critical), or it rejects the search (if the control is critical). A value of zero (which is the default) indicates that no limit should be enforced.

Defining resource limits in search requests

Each search request allows the client to specify the size limit and time limit that should be used for that operation.

If the client requests a size limit of zero, then that indicates that the client does not want to impose any limit and the search request is processed using whatever size limit imposes for that client. If the client requests a nonzero size limit and that requested size limit is lower than what the server would otherwise impose, then the client-requested size limit is used. If the client requests a size limit that is higher than what the server would otherwise impose, then the server-imposed limit is used.

The same logic applies to the requested time limit. That is, the time limit from the search request is used if it is lower than what the server would have otherwise imposed. Otherwise, the server's limit is used.

Controls for interacting with resource limits

PingDirectory Server offers a few controls pertaining to resource limits.

The get user resource limits control

The *get user resource limits request control* can be included in a bind request to indicate that the server should return information about resource limits that aree enforced for the user in the response to the successful bind. The information returned can include:

- The maximum size limit that the server imposes for the user
- The maximum time limit that the server imposes for the user

- The maximum idle time limit that the server imposes for the user
- The maximum lookthrough limit that the server imposes for the user
- The name of the client connection policy that has been selected for the connection
- The DNs of the groups in which the user is a member
- The names of the privileges that have been assigned to the user
- The DN of a user with roughly equivalent authorization characteristics that can be used for operations as the user in entry-balancing backend sets that do not contain that user

The permit and reject unindexed search controls

The *permit unindexed search request control* can be included in a search request to indicate that the server should process the requested search operation even if it is unindexed. This control is only honored if the requester has the `unindexed-search-with-control` privilege. For applications that have a legitimate need to issue unindexed search requests, this control and privilege combination can provide a better alternative than granting the requester the `unindexed-search` privilege because that privilege can allow the client to inadvertently issue unindexed search requests.

On the other hand, the *reject unindexed search request control* can be included in a search request to indicate that the server should reject the operation if the search is not at least partially indexed and therefore cannot be processed efficiently. This might help clients with the `unindexed-search` privilege avoid unintentionally requesting expensive searches.

Considerations for account security

There are several things to keep in mind when creating and managing user accounts.

Require secure communication

Whenever feasible, all communication with the server should be secure, and connections that are initially secure should be preferred over those that use StartTLS.

This can be accomplished by setting up the server with only LDAPS and HTTPS connection handlers, while leaving the LDAP and HTTP connection handlers disabled.

It is unlikely that you have clients that support StartTLS but do not support creating connections that are initially created as secure. However, if that is the case, then you should consider using the reject-insecure-requests global configuration property to ensure that clients are only allowed to establish insecure connections for the purpose of using the StartTLS extended operation.

If you do need to accept from clients that only support insecure communication, then you should try to keep those clients to a minimum, and you should take steps to limit what those clients can do when possible. Recommendations include:

- Create a client connection policy that are used for insecure connections, and use it to restrict what types of requests those clients can issue and to impose resource limits on those connections.
- Use a virtual attribute to set the `ds-auth-require-secure-communication` operational attribute to true for all users, and override that with a real attribute set to false for accounts that have a legitimate need to use insecure communication.

At a bare minimum, you should require secure communication for administrative accounts.

Prevent unauthenticated requests

Preventing requests from unauthenticated clients creates an initial hurdle that attackers must overcome for online attacks against the server. Whenever feasible, clients should be required to authenticate before they are allowed to issue requests.

If possible, use the `reject-unauthenticated-requests` global configuration property to prevent all clients from issuing unauthenticated requests. If a small, well-defined set of requests should be allowed

to unauthenticated clients, then you can use the `allowed-unauthenticated-request-criteria` property to permit them while rejecting all other types of requests.

If it is not feasible to use the `reject-unauthenticated-requests` property, then consider creating a client connection policy that matches unauthenticated connections. Use it to restrict what types of requests are allowed for unauthenticated clients and to impose significant resource limits for those clients.

Delay bind responses after too many authentication failures

You should have some mechanism in place to protect against online password guessing attacks.

Traditionally, this is done by locking accounts (at least temporarily) after too many failed authentication attempts. However, this is undesirable because an attacker could use it to intentionally lock those accounts and deny access to its legitimate owner. While you might be willing to accept this possibility for regular user accounts, you don't want to risk the chance that administrative accounts can become locked and unusable.

A compelling alternative to actually locking user accounts is to delay bind responses after too many failed attempts. This can help limit the rate at which attackers might make guesses without significantly impeding the legitimate account owner. To do this, use the `failure-lockout-action` property in the password policy configuration to select a policy that delays bind responses rather than locking the account.

If you do need to actually lock accounts to prevent them from being used after too many failed attempts, then you should choose a high enough `lockout-failure-count` value to ensure that accounts are not inadvertently locked by legitimate users who know their passwords but just mistype it several times in a row.

Require strong authentication

There are several actions that you can and should take to help ensure that passwords are not compromised.

These include:

- Use secure communication to prevent passwords (even encoded passwords) from being exposed to network observers.
- Use password validators to help prevent users from choosing weak passwords.
- Use delayed bind responses or account lockout to prevent online password guessing attacks.
- Use strong password storage schemes to resist offline password guessing attacks.
- Use data encryption, encrypted backups, and encrypted LDIF exports to ensure that encoded passwords are not accessible in the clear to anyone with access to server files.

However, using passwords as the only authentication factor might still constitute a security risk. No matter what password validation restrictions you impose, some users will still try to choose the simplest password they can get away with. There is also the risk that they will reuse passwords across multiple services, making their accounts vulnerable to credential stuffing attacks in the event of a data breach across any of those services. Phishing attacks can also trick users into revealing their passwords attackers.

To help further protect accounts in the event that their password becomes compromised, you should consider requiring strong authentication. PingDirectory Server provides three primary options for this:

- Use a SASL mechanism that supports two-factor authentication. The UNBOUNDID-TOTP mechanism is a great one that doesn't rely on interaction with any third-party service, and there are several free options to allow users to generate time-based one-time passwords. However, the UNBOUNDID-DELIVERED-OTP and UNBOUNDID-YUBIKEY-OTP mechanisms also provide much better security than just a password on its own.

  > ⓘ **Note:**
  >
  > While two-factor authentication might not completely protect against phishing attacks, because the fake site can also ask the user to provide the one-time password in addition to the static password, it does at least prevent the password from being reused.

- Consider certificate-based authentication, as with the EXTERNAL or UNBOUNDID-CERTIFICATE-PLUS-PASSWORD SASL mechanism. Certificates are much stronger than passwords when used as the only authentication factor, and requiring a certificate and password together provides the best of both worlds. Certificates are also not vulnerable to replay attacks in the way that passwords (and one-time passwords) are, and as long as clients take the proper steps to validate the authenticity of the certificate and perform hostname validation, they are much more resistant to phishing attacks.
- If users are not directly communicating with PingDirectory Server itself, but are interacting with applications that use the Directory Server behind the scenes, then the application could obtain an OAuth token for that user, and then authenticate to the Directory Server using the OAUTHBEARER SASL mechanism. This enables support for additional authentication mechanisms that are typically not available to LDAP clients, like FIDO security keys.

Even if you cannot require strong authentication for all user accounts, you should at least require it for administrators.

Use non-identifiable user DNs

It is a common practice to include identifiable information in user distinguished names (DNs).

Attributes like uid, mail, and cn are often used as RDN attributes. However, this can lead to a few undesirable issues:

- This leads to personally identifiable information in the DNs for their accounts, and DNs are susceptible to being exposed in more ways than other content in user entries. For example, DNs commonly appear in log files. Further, if a client has access control permission to see any attributes in an entry, then they can also see its DN.
- Personal information might need to change. For example, a person can change their name, username, or email address. If this information is included in an entry's DN, then it requires a modify DN rather than a modify operation to change it. This is more onerous for applications, and it can often mean that two operations are required when making such a change (a modify DN to change the values of attributes used in the entry's RDN, and a separate modify operation to change other affected attributes that are not used in the RDN). It also means that an entry's DN might not be a stable identifier over the life of that entry.
- If DNs contain identifiable information, then they also likely contain predictable information. If it is possible to make reasonable guesses about what a user's DN might be, then their account is more susceptible to targeted attacks.

A better alternative is to choose an RDN attribute that is both unpredictable and unrelated to the user. UUIDs (universally unique identifiers, as described in *RFC 4122*) are a good way to accomplish this, and PingDirectory Server already generates a UUID value for every entry stored in the server, as the value of the `entryUUID` attribute, as described in *RFC 4530*.

The `entryUUID` operational attribute type is declared with the NO-USER-MODIFICATION constraint, which means that clients are not allowed to specify its value, nor should they attempt to do so, because it is vital that it be unique across all entries in the server. This means that clients can't use it as the RDN attribute when adding new entries to the server. However, PingDirectory Server provides support for automatically using the `entryUUID` value that it generates as the entry's RDN attribute, using it in place of whatever RDN the client initially provided. There are two ways to accomplish this:

- Clients can use the *name with entryUUID request control* in add requests to explicitly request that the server replace the RDN that it provides with one based on the `entryUUID` attribute.

- The server can be configured with criteria to automatically identify which entries should be automatically created with `entryUUID` as the naming attribute. This can be done with the following properties in the global configuration:
  - `auto-name-with-entry-uuid-connection-criteria` — Connection criteria that can be used to identify clients whose add requests should automatically be updated to use `entryUUID` as the naming attribute.
  - `auto-name-with-entry-uuid-request-criteria` — Request criteria that can be used to identify which add requests should automatically be updated to use `entryUUID` as the naming attribute.

If you cannot or do not wish to use `entryUUID` itself as the naming attribute for accounts, then you might still be able to define a custom attribute for this purpose. In that case, clients would generate the value and use it as the naming attribute in add requests.

> ⓘ **Note:**
>
> It is not possible to use `entryUUID` as the naming attribute in the accounts for root users and topology administrators. These accounts reside in the configuration, and the server's configuration framework requires that `cn` be used as the naming attribute for those entries. However, we strongly recommend creating those accounts with a `cn` value that is not predictable and does not contain any identifiable information. This also applies to alternate bind DNs for those users.

Use separate accounts for each administrator

PingDirectory Server's setup process automatically creates an initial root user that can be used to manage content in the server, and it suggests a DN of `cn=Directory Manager` for this account.

Many directory servers only support creating a single root account that is shared by server administrators. However, this comes with a lot of disadvantages, including:

- If there is only a single root account, then it is difficult to audit the actions of individual administrators.
- If there is only a single root account, then its credentials have to be shared across multiple administrators. This increases the risk that those credentials will be compromised because the more people who have access to them, the greater the chance that they will be leaked.
- If there is only a single root account whose credentials have to be shared across multiple administrators, then it can be disruptive to change those credentials if required (for example, if the credentials are compromised, or if one of the administrators leaves the organization or takes on a different role).
- If there is only a single root account, then it becomes more difficult to use strong authentication for that account in a secure manner.
- If there is only a single root account, then that account must have full access to perform any operation that any administrator might need to do. If this account is shared by multiple administrators, then that can give some of them more access than necessary.

PingDirectory Server allows you to create any number of root user and topology administrator accounts. Each administrator should have their own account with separate credentials, support for strong authentication, and rights and privileges tailored to their role.

Prefer topology administrator accounts over root users

In PingDirectory Server, administrative accounts can be placed in any of three places.

- They can be created as root users. These accounts exist in the configuration (after `cn=Root DNs,cn=config`) and are not synchronized across server instances. If you want to use root accounts across multiple servers, then you must create the account in each server and keep it up to date across all of them. Root user accounts can optionally automatically inherit a default set of privileges, and you can also explicitly grant and revoke privileges as needed.

- They can be created as topology administrators. These accounts also exist in the configuration (after `cn=Topology Admin Users,cn=topology,cn=config`), and these accounts are automatically synchronized between server instances within the same topology. Topology administrators can also automatically inherit a default set of privileges, and you can also explicitly grant or revoke privileges.
- They can be created in the user data. These accounts will be replicated across all Directory Servers and they can be used to authenticate to PingDirectory and PingDirectoryProxy Servers, but they cannot be used to authenticate to PingDataSync or PingDataMetrics Servers. They cannot automatically inherit a default set of privileges, but you can explicitly grant privileges to them as needed. Accounts created in the user data can also be unavailable if the backend containing that data is offline, such as when performing an online restore, replica initialization, or LDIF import.

We recommend using topology administrator accounts over root users or accounts created in the user data. Topology administrators have all of the same capabilities as root users, and their accounts are also automatically synchronized across all servers in the topology so there is no need to apply the same change across multiple servers.

See the `config/sample-dsconfig-batch-files/create-topology-admin-user.dsconfig` batch file for more information about creating topology administrator accounts.

Disable or delete the initial root account

The initial root user account that `setup` creates should only be used to apply an initial set of configuration changes and create individual accounts for all of the other administrators.

From that point on, each administrator should use their own account for managing the server, and the initial root account is no longer needed.

To ensure that the initial root user account cannot be compromised or otherwise used inappropriately, it should be disabled by setting its `disabled` property to true or by setting the `ds-pwp-account-disabled` operational attribute to true in the configuration entry or completely removed from the server.

See the `config/sample-dsconfig-batch-files/disable-or-remove-the-initial-root-user.dsconfig` batch file for more information about disabling or removing this account.

Logging

Logging is the best mechanism for understanding what is happening in the server.

Log messages can be invaluable when troubleshooting problems, and log analysis can reveal performance characteristics and usage patterns.

Types of loggers

PingDirectory Server offers several types of loggers. It allows you to have any number of loggers of each type.

Error loggers

Despite their name, error loggers are not only used to report about errors. They can also provide information about significant events that occur within the server as well as warning conditions that might become a problem in the future.

Error loggers usually do not report information about operation processing unless an internal error is encountered during the course of processing an operation. Operation processing is covered by access loggers.

Each error log message has a severity that can be used to indicate how important the message is. Defined severities include:

- Fatal Error — Critical problems that might affect the server's ability to continue running
- Severe Error — Significant problems that might affect the server's functionality

- Mild Error — Less significant and more isolated problems that might not require immediate attention
- Severe Warning — Warnings about significant issues that might not represent errors can still require administrative action
- Mild Warning — Warnings about less significant issues
- Notice — Informational messages about significant events that occur within the server
- Information — Informational messages about less significant events
- Debug — Information that can generally only be useful when debugging low-level problems

Each logger can be configured to indicate which message severities should actually be recorded. The text-based error logger that is enabled in the out-of-the-box configuration logs messages at the following severities by default: fatal error, severe error, severe warning, and notice.

Each error log message also includes a category that is related to the server subsystem that generated it. Loggers can optionally be configured to record messages with different severities on a per-category basis.

PingDirectory Server offers several types of error loggers:

- The text error log publisher writes messages in a structured plain-text format to files on the server filesystem. The default error logger that the server maintains is a text error log publisher. The UnboundID LDAP SDK for Java provides an API for parsing error log messages written by this log publisher.
- The JSON error log publisher writes messages as JSON objects to files on the server filesystem. These messages are more likely to be parsable by third-party software than the format used by the text error log publisher.
- The console JSON error log publisher writes messages as JSON objects to the server's standard output or standard error stream. This is primarily useful for cases in which the server is running in Docker or some other type of container, and when using a framework that consumes standard output and standard error content for ingestion into a log management system.
- The syslog error log publisher writes messages to a syslog server. This can allow log messages to be sent to a centralized system through a commonly used logging protocol.
- The Java Database Connectivity (JDBC) error log publisher writes messages to a relational database using a specified column layout. This can also allow for centralized logging.

You can also use the UnboundID Server SDK to create custom error loggers. These can be used to customize the format of the log messages or to send log messages to other types of systems.

Access loggers

Access loggers are used to record information about interaction with clients. This includes:

- New connections that are established
- Connection closure and termination
- TLS negotiation used to secure connections, including certificate chains presented by clients
- Requests issued by clients
- Results returned by the server, including entries and references returned for searches as well as intermediate responses returned for any operation
- Requests that are forwarded to other servers
- The result of any replication assurance processing that was performed
- Entry rebalancing processing performed by PingDirectoryProxy Server to move data between backends sets

Access log messages should always include a connection ID that identifies the associated client connection. Messages that are associated with operations (those other related to connects, disconnects, and TLS negotiation) should also include an operation ID that can be used to identify all log messages associated with that operation.

Access loggers might not actually record all of these types of messages. For example, the server's default access logger is configured to only record one message per operation when processing has completed

for that operation. That message includes details about the request and result. Search result messages include the number of entries and references for that operation, but it does not record a separate message for each entry and reference that is returned.

PingDirectory Server also provides support for filtered logging. You can define fine-grained criteria to indicate which types of messages should be included in the log. This can be used to create log files that only include certain types of messages or to exclude messages that you consider unimportant.

The types of access loggers that PingDirectory Server offers include:

- The text access log publisher writes messages in a structured plain-text format to files on the server filesystem. The default access logger that the server maintains is a text access log publisher.The UnboundID LDAP SDK for Java provides an API for parsing access log messages written by this log publisher.
- The JSON access log publisher writes messages as JSON objects to files on the server filesystem. These messages are more likely to be parsable by third-party software than the format used by the text access log publisher.
- The console JSON access log publisher writes messages as JSON objects to the server's standard output or standard error stream. This is primarily useful for cases in which the server is running in Docker or some other type of container and when using a framework that consumes standard output and standard error content for ingestion into a log management system.
- Text audit log publishers write information about changes processed by the server to files using the standard LDIF format. Audit logs can be helpful in understanding exactly what changes have been made to data in the server, and they can also be useful if you need to replay or revert changes.
- The operation timing access log publisher is similar to the text access log publisher, but it can also include detailed information about the length of time required to process various phases of an operation. This can help identify which portions of operation processing are most expensive and might be able to help identify potential ways to improve performance.
- The debug access log publisher is a file-based logger that writes multi-line messages with detailed information about the contents of each request received from a client and response returned by the server. It can optionally be configured to also include detailed information about access control evaluation performed in the course of processing the operation.
- The syslog access log publisher writes messages to a syslog server. This can allow log messages to be sent to a centralized system through a commonly used logging protocol.
- The JDBC access log publisher writes messages to a relational database using a specified column layout. This can also allow for centralized logging.
- The admin alert access log publisher can be used to generate an administrative alert whenever the server processes an operation that matches the associated logging criteria.

The UnboundID Server SDK also provides support for creating custom access loggers if you want to use a particular log format or write log messages to other targets.

PingDirectory Server also offers the following JSON-formatted loggers:

JSON-formatted audit loggers

These complement the existing file-based audit logger in that they provide a record of changes to data in the server. Whereas the existing file-based audit logger writes information about changes as LDIF records, these loggers record information about the changes as JSON objects. There are three flavors of these JSON-formatted audit loggers:

- One that writes to files
- One that writes to the console (standard output or standard error)
- One that writes to syslog

JSON-formatted HTTP operation loggers

These complement the existing HTTP operation logger in that they provide a record of all HTTP requests received by and responses sent by the server. These messages are also formatted as JSON objects, and there are file-based, console-based, and syslog-based versions.

JSON-formatted Sync loggers

These complement the existing file-based Sync logger and Sync failed operations loggers in that they provide information about the processing performed by the Synchronization Server, formatted as JSON objects. Again, these log messages can be written to files, to the console, or to syslog.

HTTP operation and trace loggers

HTTP operation loggers record information about requests and responses to HTTP-based clients. PingDirectory Server provides two types of HTTP operation loggers:

- A file-based logger that uses the W3C common log format, which includes a timestamp, client address, the username of the requester, the HTTP request method, the request URI and optional query string, the protocol, and the status code.
- A file-based logger that provides more detailed information about each request, which can include elements like request headers, the authorization type, cookie names, request parameters, the request content type, and the response content length.

You can also use the UnboundID Server SDK to customize HTTP operation loggers. Trace loggers are also used for recording information about processing associated with HTTP clients, but they can include content that is more specific to the type of request being processed, including OAuth token validation, SCIM, the Directory REST API, and the consent API.

PingDirectory server provides the following types of trace loggers:

- A file-based logger that writes information in plain text format.

Sync loggers

Sync loggers provide a way to record information specific to processing performed by PingDataSync Server. Events that can trigger a sync log message include:

- Whenever a change is detected from a source
- Whenever a change is applied at the destination
- Whenever an attempt to apply a change fails for some reason
- Whenever a change is dropped or ignored (for example, because it was not needed or was outside the scope of all configured sync classes)
- Whenever a source entry is mapped to a destination entry
- Whenever a change is aborted during plugin processing
- Whenever a plugin generates a custom log message

The following types of sync loggers are available:

- A text sync log publisher that uses a multi-line plain text format to record information about PingDataSync Server processing
- A text sync failed ops log publisher that uses a mutli-line plain text format to record more detailed information about failed synchronization processing

ⓘ **Note:**

Both of these log publishers generate output that is intended to be read by a person. It is not optimized for automated parsing.

Debug loggers

Debug loggers provide a way to obtain detailed information about internal processing performed by the server. This can include specific informational content specifically added for low-level debugging, but at a minimum it likely includes things like any exceptions caught and handled internally during processing.

PingDirectory Server offers the following types of debug loggers:

- A file-based debug log publisher that records messes about debug content generated within the server. These messages can span multiple lines, and they are primarily intended to be read by people rather than parsed by automated mechanisms.

Debug logging has the potential to be extremely verbose, and it might have a notable impact on performance. It should only be enabled when it is needed, and it should be limited to as small an area of the server as possible. Because optimal configuration can require detailed knowledge of server internals, debug logging should generally only be used under the direction of support personnel.

However, there is one area of debugging that can be useful without assistance from support: debugging issues with custom extensions. The UnboundID Server SDK provides support for generating debug messages within custom extensions, and PingDirectory Server includes a “Server SDK Extension Debug Logger” instance that is pre-configured to record debug messages generated by extensions created using the Server SDK.

Standard output logger

The server should only write data to standard output or standard error under very limited circumstances, like when it is specifically configured to write JSON-formatted access or error log messages to the console. However, there might be limited cases in which it could write to standard output or standard error for other purposes, or in which the underlying JVM generates messages written to those streams (for example, if you enable certain low-level JVM debugging features). In such cases, the output that would have been written to standard output or standard error should be redirected into the `logs/server.out` file.

Log file rotation and retention

PingDirectory Server provides support for rotating log files to limit how large they can become. It also provides support for deleting old log files to limit how much total disk space can be consumed by log content.

Rotation policies

Each file-based logger can be configured with one or more rotation policies that can be used to determine when the server should rotate the active log file. When this occurs, it closes the active log file, renames it to include a timestamp indicating when it was rotated and creates a new, empty file to use for subsequent log messages.

PingDirectory Server provide support for the following log rotation policies:

- Size-Based — Rotate log files after they reach a specified size.
- Time Limit — Rotate log files after the active file has been in use for at least a specified length of time.
- Filed Time — Rotate log files at fixed times of the day.
- Never Rotate — Do not rotate the file. This should only be used in limited circumstances because it can allow log files to grow large and potentially consume all available disk space.

If a logger is configured with multiple rotation policies, then a log file can be rotated as soon as it satisfies the criteria for any of those policies. For example, if a logger is configured to rotate files once they reach 100 megabytes or after they have been active for 24 hours, then a log file can be rotated in less than 24 hours if it grows larger than 100 megabytes within that time frame. Conversely, it can be rotated before it reaches 100 megabytes in size if it does not hit that size limit within 24 hours.

Log file rotation listeners

Whenever a logger rotates its active log file, it can optionally perform additional processing on that file. This can be accomplished through a log file rotation listener. PingDirectory Server provides the following log file rotation listener implementations:

- Copy Log File — Copies the log file to a specified alternative location.
- Summarize — Invokes the *summarize-access-log* tool on the rotated log file.

You can also use the UnboundID Server SDK to create custom log file rotation listeners.

Retention policies

Retention policies are used to determine when the server should delete a log file that had previously been rotated.

PingDirectory Server supports the following retention policies:

- File Count — Only retain a specified number of rotated log files.
- Time Limit — Only retain rotated log files that are younger than a specified age.
- Size Based — Only retain a specified maximum aggregate size of rotated log files.
- Free Disk Space — Delete rotated log files if needed to maintain a specified minimum amount of free disk space.
- Never Delete — Never delete rotated log files. This should only be used in limited circumstances because it can allow rotated log files to accumulate and potentially consume all available disk space.

If a logger is configured with multiple retention policies, then rotated log files can be deleted if the criteria associated with any of the policies is reached. For example, if a logger is configured to maintain up to twenty rotated files or up to one gigabyte of disk space, then it can retain fewer than twenty files if the aggregate space consumed by rotated log files exceeds one gigabyte. Alternatively, it can retain less than a gigabyte of rotated log files if the most recent twenty files consume less than one gigabyte of space.

When the logger needs to delete one or more files because of the retention policy, it deletes the files in chronological order based on the timestamp in the rotated file name, starting with the oldest timestamp first.

Filtered logging

PingDirectory Server allows you to access loggers with criteria that are used to identify which messages should actually be recorded.

This allows you to:

- Maintain a log containing only non-successful operations.
- Maintain a log containing operations that took longer than a specified duration to complete.
- Maintain a log of requests by root users or topology administrators.
- Maintain a log of search operations in which encoded passwords were returned to clients.

Enable filtered logging using the following access logger configuration properties:

`connection-criteria`

An optional connection criteria object that is required to match the associated client connection. If connection criteria is provided, then the logger does not attempt to generate any connect, disconnect, request, search entry, search reference, or result log messages from clients that do not match that criteria.

`request-criteria`

An optional request criteria object that is required to match the associated operation request. If request criteria is provided, then the logger does not attempt to generate any request, search entry, search reference, or result log messages for requests that do not match that criteria.

result-criteria

An optional result criteria object that is required to match the associated operation result. If result criteria is provided, then the logger does not attempt to generate any result log messages for results that do not match that criteria.

search-entry-criteria

An optional search entry criteria object that i required to match the associated search result entry. If search entry criteria is provided, then the logger does not attempt to generate any search result entry log messages for entries that do not match that criteria.

search-reference-criteria

An optional search reference criteria object that is required to match the associated search result reference. If search reference criteria is provided, then the logger does not attempt to generate any search result reference log messages for references that do not match that criteria.

See the `log-operations-by-administrators.dsconfig` and `log-encoded-password-access.dsconfig` batch files in the `config/sample-dsconfig-batch-files` directory for examples of creating loggers that use criteria to filter log messages.

Log file compression

PingDirectory Server supports compressing log files, which can help significantly reduce the amount of disk space required to store log files on disk.

Because the server does not support maintaining a mix of compressed and uncompressed content for the same log, it can only be enabled when creating a new logger. When creating a new logger in which you want to compress the content, use the following configuration property:

compression-mechanism

Indicates the type of compression that should be used when writing log files. Available values include:

- `none` — Do not use compression for the log files.
- `gzip` — Compress the log files using the gzip compression algorithm.

Because PingDirectory Server allows you to create any number of loggers of the same type, you might want to maintain both compressed and uncompressed versions of the same content. The compressed version of the logger can have rotation and retention policies allowing for keeping a larger amount of log data for a longer period of time. The uncompressed version of the logger can keep a smaller amount of log content so that it is easier to find information about recent client interaction or other content without needing to decompress any files. However, both the `search-logs` and `summarize-access-log` tools transparently support operating on compressed log files.

Log file encryption

PingDirectory Server also supports encrypting log files, which can help prevent unauthorized access to sensitive information that they might contain.

As with compressed logging, the server cannot maintain a mix of encrypted and unencrypted content for the same log, so you can only enable encryption when the logger is created. This can be done using the `encrypt-log` configuration property. See the `config/sample-dsconfig-batch-files/create-encrypted-loggers.dsconfig` batch file for an example that demonstrates the process for creating encrypted access and error loggers. These examples also enable compression for the log files.

Log parsing APIs

The UnboundID LDAP SDK for Java provides support for programmatically parsing log file content.

This includes:

- Use the *AccessLogReader* class to read and parse access log messages using the server's default text-based access log format.
- Use the *ErrorLogReader* class to read and parse error log messages using the server's default text-based error log format.
- Use the *AuditLogReader* class to read and parse change records written by the server's audit logger. While the audit logger writes changes in LDIF form (and can therefore also be read by the *LDIFReader*), the AuditLogReader provides additional support for extracting information from comments that are associated with log messages, including:
  - The message timestamp
  - The connection ID
  - The operation ID
  - The requester distinguished name (DN) and IP address
  - The entry DN for any alternate authorization identity that was used
  - The replication change ID
  - For add operations, whether the operation was an undeleted
  - For delete operations, the attributes from the deleted entry, whether it was a soft delete, or whether it was a subtree delete
  - For modify operations, whether the operation targeted a soft-deleted entry
- Use the *JSONObjectReader* class to files comprised of JSON objects. Use this to read the contents of JSON-formatted access and error log files.

Logging Tools

PingDirectory Server provides tools that can be used to access content in access and error log files.

`search-logs`

Use the `search-logs` tool to search for content in log files. This tool provides grep-like support for searching log files, but it offers the a number of additional benefits, including:

- It can automatically trace backward through rotated log files to find matching records in older log files.
- It supports searching log files that are compressed and encrypted.
- It can handle multi-line messages.
- It allows you to specify start and end times for the messages to match.

`summarize-access-log`

Use the `summarize-access-log` tool to examine one or more access log files and produce a plain-text report of the log data that they contain. The output can include:

- The length of time covered by the log files that were examined
- The number of connections that were established and disconnected
- The addresses of the clients that most frequently connected to the server
- The average rate of connects and disconnects per second
- The most common TLS protocols and cipher suites
- The number of operations processed, both overall and by operation type
- The average rate of operations processed per second, both overall and by operation type
- The average duration of operations processed, both overall and by operation type
- The breakdown of operation processing times into sets of predefined buckets, ranging from less than one millisecond to over one minute
- A breakdown of the most common result codes for each type of operation and their relative frequencies
- The most common authentication mechanisms
- The most common bind distinguished names (DNs) for successful and failed bind attempts
- The most common types of extended operations processed and their relative frequencies

- The number of unindexed search operations processed and the most common types of filters used when processing unindexed searches
- The most common base DNs for searches with non-baseObject scopes
- The relative frequencies for each search scope
- The most common types of search filters used and their relative frequencies
- The most common types of filters for searches returning zero, one, and multiple entries
- Filters used for searches that took the longest to complete

The `summarize-access-log` tool supports operating on log files that are compressed and encrypted. It also attempts to anonymize sensitive information in the output by replacing attribute values with placeholders.

Change logging

PingDirectory Server can be configured to maintain an LDAP-accessible changelog with a record of changes that have been processed in the server.

It is based on the specification in *draft-good-ldap-changelog*, but includes a number of proprietary enhancements that provide access to additional useful information. This can include:

- The values of updated attributes as they appeared before and after the change
- The values of a configured set of key attributes from the entry, even if they weren't altered by the change
- The content of an entry that was deleted
- The values of virtual attributes from the entry

The changelog can be useful for auditing changes that have been processed in the server, as well as for synchronizing changes to other systems. It is disabled by default, but it can be enabled with the following configuration change:

```
dsconfig set-backend-prop \
     --backend-name changelog \
     --set enabled:true
```

Additional properties that you might want to use to customize the changelog configuration include the following.

Property	Description
`changelog-include-attribute`	Specifies which attributes are included in changelog entries for add and modify operations. If this is specified, then only those attributes are included, even if the operation added or updated other attributes.
`changelog-exclude-attribute`	Specifies which attributes should be excluded from changelog entries for add and modify operations. By default, the changelog excludes a number of attributes that might contain sensitive information that is unlikely to be required externally. However, encoded passwords are not excluded by default because it might be necessary to synchronize them to other systems.
`changelog-deleted-entry-include-attribute`	Specifies which attributes should be included in changelog entries for delete operations. By default, all user attributes are included, but operational attributes are not.

Property	Description
`changelog-deleted-entry-exclude-attribute`	Specifies which attributes should be excluded from changelog entries for delete operations.
`changelog-include-key-attribute`	An optional set of attributes that should be included in changelog entries, even if they were not changed in the course of processing the operation.
`changelog-max-before-after-values`	Indicates that the changelog entry should include up to the specified number of before and after values for the updated attributes. This is zero by default to indicate that before and after values should not be included, but if it is changed to a nonzero value, then before and after values are included for any changed attributes whose value count is below this limit.
`use-reversible-form`	Indicates whether to log modifications in reversible form, which contains enough information to allow the change to be reverted. By default, changes are logged using the set of modifications as the client requested them.
`include-virtual-attributes`	Indicates which types of virtual attributes should be included in changelog entries. This might include zero or more of the following: ▪ `add-attributes` — Indicates that the changelog entry for an add operation should include any virtual attributes that would be generated for the entry at the time it was added. ▪ `deleted-entry-attributes` — Indicates that the changelog entry for a delete operation should include any virtual attributes present in the entry at the time it was deleted. ▪ `before-and-after-values` — Indicates that the changelog entry should include any virtually values for attributes updated in the operation. ▪ `key-attribute-values` — Indicates that the changelog should include virtual values for any key attributes to be included.
`apply-access-controls-to-changelog-entry-contents`	Indicates whether the server should pare down the contents of each changelog entry based on the requester's access control rights for the updated entry.

Property	Description
`report-excluded-changelog-attributes`	Indicates whether to report information about any attributes that were excluded from the changelog entry on the basis of the `apply-access-controls-to-changelog-entry-contents` property. Allowed values include: ▪ `none` — Do not report on any attributes that were excluded. ▪ `attribute-counts` — Report the number of attributes that were excluded. ▪ `attribute-names` — Report the names of the attributes that were excluded.
`soft-delete-entry-included-operation`	Indicates whether to include operations that target soft-deleted entries. By default, operations that target soft-deleted entries, but they can be included with one or more of the following values: ▪ `delete` — Indicates that the changelog should include records of soft-deleted entries that are removed from the server. ▪ `modify` — Indicates that the changelog should include records of modify operations that update soft-deleted entries.

By default, the server does not include any access control rules that grant users access to retrieve changelog entries. As such, only users with the `bypass-acl` or `bypass-read-acl` can see them. If you want to grant access to other users who are subject to access control evaluation, you must do so using global ACIs.

The data recovery log

PingDirectory Server is preconfigured with an audit log instance that can be used to help replay or revert changes if the need arises. This is the data recovery log, and the log files are written into the `logs/data-recovery` directory.

The log files are compressed, and they are also encrypted if data encryption is enabled in the server. The logger is configured to use reversible form to make it usable to revert changes as well as to replay them. The server is configured to keep the data recovery log files for up to one week or up to ten gigabytes of disk space.

The server also provides an `extract-data-recovery-log-changes` tool to use in conjunction with this log. It offers a wide variety of arguments that can be used to identify which changes to extract, and it generates an LDIF file with the extracted changes. The changes can be exported in a form that allows them to be replayed, or they can be exported in a form that allows them to be reverted.

Some of the things that can be used to identify the changes to extract from the data recovery log are:

- The time that the changes were processed
- The operation types (add, delete, modify and modify distinguished name (DN)) for the changes to extract
- The connection ID for the connection on which the changes were requested
- The identity of the user that requested the changes
- The location of the changes in the DIT
- The content of the change

Monitoring

Monitoring is a vital part of a deployment because it allows you to ensure that the service remains available and functioning properly and because it can help alert you to problems as soon as they arise. PingDirectory Server provides a broad range of monitoring support.

Monitor entries

PingDirectory Server provides a monitor backend, based at `cn=monitor`, with a wealth of information about the current state of the server.

Some of the available monitor entries include:

- Information about the version of the server and key libraries that it uses
- A list of all client connections currently established to the server
- A list of all active operations currently being processed in the server
- Information about each connection handler that is enabled in the server
- Information about each backend that is enabled in the server
- Information about each database used by local DB backends
- Information about the overall database environment for each local DB backend
- Information about the server's replication state
- Information about external servers accessed by PingDirectoryProxy Server
- Information about load balancing state in PingDirectoryProxy Server
- Information about entry balancing state in PingDirectoryProxy Server
- Information about PingDataSync Server's sync engine and pipes
- A timeline of server state changes
- Information about the host system on which the Java Virtual Machine (JVM) is running
- Information about the work queue and worker thread utilization
- Information about JVM memory utilization
- Information about file system disk space utilizationInformation about HTTP servlets configured within the serverInformation about cache utilization
- Information about each certificate that is in use within the server
- Information about the name resolution performance
- Information about server file descriptor usage and availability
- Stack traces for all active threads in the server
- Information about result codes returned by different types of operations
- Information about processing times, both overall and on a per-operation basis, including a histogram of processing times
- Information about supported and enabled TLS protocols and cipher suites
- A summary of various metrics that can be used for determining the availability status of the server
- Detailed information about the JVM and environment in which the server is running

You can also use the UnboundID Server SDK to create custom monitor entries. While this is typically done for the purpose of exposing monitor information about other custom extensions, it could also be used for other purposes, like providing information about external systems or services.

Each monitor entry is a read-only entry that can be searched and retrieved like data in user backends. However, the server does not maintain indexes for arbitrary monitor attributes. For best performance when searching for and retrieving monitor entries, you should do one of the following:

- If you know the DN of the desired monitor entry, issue a search with that DN as the search base DN and a scope of baseObject. In this case, any filter can be used.
- If you do not know the DN of the monitor entry, or if you want to retrieve multiple entries of a given type, then issue a search with a base DN of `cn=monitor`, a whole Subtree scope, and a filter that is based on the specific structural object class for that type of monitor entry (for example, `(objectClass=ds-`

`backend-monitor-entry)` if you want to retrieve entries providing general information about each backend enabled in the server). The filter can optionally be ANDed with other components to further narrow down the search results.

Any LDAP client can be used to retrieve these monitor entries, but the UnboundID LDAP SDK for Java provides specific support for many types of monitor entries.

The availability state servlet

There are cases in which you want to obtain basic information about the state of the server over HTTP, or ideally, HTTPS.

These might include:

- When server HTTP endpoints are fronted by a load balancer capable of issuing and interpreting the results of HTTP requests for health-checking purposes
- When using an orchestration framework like Kubernetes and you want to assess the current state of the server to determine if the instance is in a state that is ready to be used or if it has encountered an issue that has caused it to become degraded or unavailable

To assist with this, PingDirectory Server includes an availability state HTTP servlet that can be used to determine whether the server considers itself to be available, degraded, or unavailable. It is accessed using the GET, POST, or HEAD methods, and it returns a configurable status code and optional JSON-encoded response body that can be used to determine the server's self-assessed health state.

There are two instances of this servlet configured by default:

- One that uses a path of `/available-state` that returns an HTTP status code of 200 (OK) if the server considers itself available or an HTTP status code of 503 (Service Unavailable) if the server considers itself degraded or unavailable
- One that uses a path of `/available-or-degraded-state` that returns an HTTP status code of 200 if the server considers itself available or degraded or an HTTP status code of 503 if the server considers itself unavailable

The server's assessment of its health state is based on the presence of any unavailable or degraded alert types that are active in the server. If the server currently has one or more unavailable alert types, then it considers itself unavailable. If the server does not have any unavailable alert types but has one or more degraded alert types, then it considers itself degraded. If the server does not have any unavailable or degraded alert types, then it considers itself available.

Administrative alerts

PingDirectory Server can generate administrative alerts to notify administrators of errors, warnings, and significant events that occur in the server. Each alert has a name, severity, and message.

The `docs/admin-alerts-list.csv` and `docs/admin-alerts-list.html` files provide a listing of all alert types that are defined in PingDirectory Server. Some of these alert types include:

- Whenever the server begins or completes the startup process
- Whenever the server begins the shutdown process
- Whenever a change is made to the server configuration
- If the server detects that a configuration change was made with the server offline
- Whenever a change is made to the server's access control policy
- If a backend is disabled with the server online
- If an error occurs while attempting to enable a connection handler
- If an error occurs while attempting to enable a backend
- If an error occurs while trying to create or restore a backup
- If an error occurs while trying to retrieve or decode an entry from a backend database

- If the server detects that a backend was not cleanly shut down and it can take additional time to bring it back up while it performs recovery processing

  If a backend index needs to be built or is in the process of being built
- If replication has become backlogged, is missing changes, fails to replay a change, encounters a conflict that cannot be automatically resolved, or encounters some other problem
- If the server work queue has become backlogged or full
- If an attempt is made to assign a user an invalid privilege
- If available disk space becomes too low
- If debug logging is enabled
- Whenever the server executes a command on the underlying system
- Whenever the server enters or leaves lockdown mode
- Whenever an alarm is raised or cleared
- If the server detects an invalid or non-recommended Java Virtual Machine (JVM) configuration
- If PingDirectoryProxy Server changes the assessed health check state for any of its backend servers
- If PingDirectoryProxy Server encounters a problem while performing entry rebalancing processing
- If PingDataSync Server detects a backlog or encounters a problem during processing

Whenever an administrative alert is raised within the server, the server provides it to all alert handlers that are defined and enabled in the configuration. Alert handlers that are available in PingDirectory Server include:

- Error Log — Write a message about the alert to server error log.
- Exec — Execute a specified command on the underlying system.
- JMX — Emit a Java Management Extensions (JMX) notification with information about the alert.
- Output — Write a message about the alert to standard output or standard error.
- SMTP — Send an email message with information about the alert to a specified set of recipients.
- SNMP — Emit an SNMP trap with information about the alert.
- Twilio — Send an SMS text message with information about the alert using the *Twilio* service.

The UnboundID Server SDK can also be used to create custom alert handler implementations.

Information about recent alerts generated by the server is also available in the `cn=alerts` backend. It is also included in the output of the `status` tool.

Alarms and gauges

Each PingDirectory Server includes a set of gauges for tracking various metrics over time.

Each gauge is associated with a monitor attribute and can be configured with warning, minor, major, and critical thresholds. If the monitor value crosses one of those threshold values, then the server raises an alarm condition that might indicate a problem such as low disk space or unavailability of an external server, and can optionally trigger administrative alerts.

Like alerts, alarms have a name, severity, and message. They also have a condition that is indicated by the alarm, like “Disk Usage,” and they can be associated with a specific resource, such as which filesystem is running low on disk space. If alarm conditions are published using SNMP, then they can also include probable cause and alarm type properties.

Alarms are exposed in the `cn=alarms` backend. Their values can change over time, and they can be cleared whenever the monitored value returns to normal. Active alarms can be viewed using the `status` tool.

Account status notifications

PingDirectory Server can generate account status notifications whenever certain events occur in the server regarding an account's state.

These notifications are used to notify end users or administrators about that event, or potentially to invoke custom code in response to them. For more information, see the *Account status notifications* section in the discussion on password policies.

Stats logging

PingDirectory Server provides a stats logger plugin that can be used to write a log file with a variety of performance metrics and other information collected within the server.

The information can be written in either CSV or JSON formats. Although CSV was originally the only format supported for this log file, use JSON for new deployments because it is easier to parse in automated form, especially given that the set of fields included in the output can change over time.

A configuration change like the following can be used to enable a stats logger instance using the JSON format.

```
dsconfig create-plugin \
  --plugin-name "JSON-Formatted Stats Logger" \
  --type periodic-stats-logger \
  --set enabled:true \
  --set log-file:logs/stats-log.json \
  --set log-file-format:json \
  --set "rotation-policy:Fixed Time Rotation Policy" \
  --set "rotation-policy:Size Limit Rotation Policy" \
  --set "retention-policy:File Count Retention Policy" \
  --set "log-interval:1 s" \
  --set "collection-interval:200 ms" \
  --set suppress-if-idle:true \
  --set local-db-backend-info:basic \
  --set replication-info:basic \
  --set included-ldap-stat:active-operations \
  --set included-ldap-stat:num-connections \
  --set included-ldap-stat:op-count-and-latency \
  --set included-ldap-stat:work-queue \
  --set included-resource-stat:memory-utilization \
  --set histogram-op-type:all
```

External monitoring

While PingDirectory Server provides extensive access to monitoring information to clients or in files on the server filesystem, there is a lot of benefit to using an external mechanism for monitoring individual servers and especially for aggregating information across multiple servers.

PingDataMetrics Server

PingDataMetrics Server collects and aggregates performance and event data from a set of PingDirectory, PingDirectoryProxy, and PingDataSync Server instances. It can report the overall performance of the entire directory service, as well as of individual servers. PingDataMetrics Server normalizes and aggregates this data and makes it available through a REST API. It also generates charts for viewing the information in PingDataMetrics Server's web interface, and both historical and current metrics are available.

StatsD endpoints

PingDirectory Server offers support for StatsD endpoints that can send metrics to third-party monitoring software using a simple, well-defined protocol. Many popular monitoring products (like Splunk and DataDog) provide support for ingesting metrics using the StatsD protocol.

PingDirectory Server supports communicating with StatsD servers over TCP or UDP. It optionally supports TLS encryption when using TCP-based communication.

JMX

Java Management Extensions (JMX) is a core java framework that provides support for publishing metrics and notifications. JMX is supported by a wide range of monitoring software, and the `jconsole` tool that is provided as part of Java installations can also be used to interact with JMX-enabled services.

The PingDirectory Server provides support for publishing all monitor information as JMX MBeans. It also provides support for a JMX alert handler that can generate a JMX notification in response to administrative alerts that are raised within the server. The `jmx-read` privilege is required for access to monitoring data, and the `jmx-notify` privilege is required to be able to subscribe to JMX notifications.

SNMP

SNMP is another standard protocol that is widely supported by monitoring software. PingDirectory Server can act as an SNMP subagent to make selected monitoring information available for consumption by SNMP clients and monitoring software. The server can also generate SNMP traps in response to administrative alerts that are raised within the server.

Auditing

It is important to regularly audit PingDirectory Server and its content to ensure that the data remains secure.

Auditing configuration changes

Proper server configuration is critical to maintaining security. Be aware of all changes to the server configuration and understand whether the configuration in its current state matches what you intend and expect it to be.

Administrative alerts

PingDirectory Server generates an administrative alert whenever a configuration change is made with the server online. It also generates an alert during startup if it detects unauthorized changes to the `dsconfig` tool in offline mode or the `manage-profile` tool.

You should make sure that an appropriate set of alert handlers are in place so that administrators are notified of configuration changes as soon as they occur.

The configuration audit log

PingDirectory Server maintains a `logs/config-audit.log` file that contains a record of all authorized configuration changes made within the server. This includes:

- Changes made through the `dsconfig` command-line tool with the server online
- Changes made through the `dsconfig` command-line tool running in offline mode
- Changes made through the web administration console
- Changes made through the `manage-profile` tool
- Changes made through the configuration API
- Changes made by clients updating configuration entries over LDAP

The configuration audit log will not include a record of any changes made by directly editing the `config.ldif` file with the server offline, but the server should detect any such changes at startup and generate an administrative alert in response to them.

Each record in the configuration audit log should include the following information:

- A timestamp indicating when the change occurred
- The connection ID and operation ID for the request that was used to make the change
- The DN of the user who made the change and the type of authentication they used
- The address of the client system used to request the change
- A command that is used to undo the change
- The change that was applied

The configuration archive

PingDirectory Server also maintains a configuration archive, in the `config/archived-configs` directory. This directory should contain a compressed and timestamped copy of every version of the configuration that the server has used. It also includes a version of the configuration as it existed when setup completed and a “clean” baseline configuration for the current version of the server without any customization applied.

The `config-diff` tool

PingDirectory Server provides a `config-diff` tool to compare different versions of the server configuration and identify differences between them. This tool can compare different versions of the configuration from the same server, or it can be used to compare configurations between different servers. Any differences will be written in the form of a `dsconfig` batch file that can update the source server so that its configuration matches that of the target.

Auditing data access

It’s also important to understand the kinds of requests that clients make. This can help you identify requests that are out of the ordinary, which might be evidence of a potential attack.

Log analysis

The primary way to accomplish this is through log analysis. It is important to regularly examine (or at least summarize) the log files to classify the types and frequency of operations being performed. A sudden increase in any one kind of traffic can be a red flag, but situations like the following might be of particular interest:

- Connections from unusual client addresses, or a spike in requests from certain clients.
- Operations that fail with invalidCredentials (49) or insufficientAccessRights (50) results.
- Search operations that fail with a timeLimitExceeded (3), sizeLimitExceeded (4), or adminLimitExceeded (11) results.
- Search operations that do not return any entries.
- Search operations that return large numbers of entries.
- Search or compare operations that explicitly attempt to retrieve or target the userPassword attribute (or another password attribute).
- Search operations that might represent attempts to exfiltrate data from the server through trawling. This can include things like repeated substring filters with subInitial filters in sequential order, such as searches with filters like `(cn=aa*)`, `(cn=ab*)`, `(cn=aa*)`, and so on. It can also include greater-or-equal and less-or-equal filters with similar patterns.
- Unindexed search attempts.

If you regularly characterize the types of operations that clients request, then you might be able to more easily identify anomalous requests.

There might be other specific events that you always want to know about. For example, you might want to know whenever clients retrieve search result entries that contain encoded passwords, or whenever a client alters the server’s access control definitions or the set of privileges granted to an account. In such cases, you can create criteria to specifically identify those types of operations, and then you can use that criteria to create a logger that only records those types of events.

If you want to know about these kinds of events right away, then you could create an admin alert access logger using that criteria. This causes the server to generate an administrative alert (with an alert type of `access-log-criteria-matched`) whenever it processes an operation matching that criteria.

Account status notification handlers

You might want to audit certain events related to password policy processing. For example, if the server is configured to lock accounts after too many failed attempts, then you might want to have a record of any time that happens. You might also want to have a record of all administrative password resets and self password changes.

This can be accomplished with account status notification handlers, and the server provides implementations that can either record messages about these kinds of events in the server error log or that can send email messages about them. If you would like to handle them in other ways, then you can use the UnboundID Server SDK to create a custom account status notification handler that implements the desired behavior.

Auditing data content

It's also a good idea to keep track of the data itself and identify any potential security-related issues that affect user entries or other data.

Data security auditors

One way to do this would be to regularly export the data to LDIF (ideally in encrypted form so that it is not exposed in the clear) and examine its content for unusual or undesirable conditions. You could also do this with search operations. However, in large data sets, such searches can be expensive and time consuming.

The server also provides an "audit data security" administrative task and an associated audit-data-security command-line tool that can help with this. When invoked, the task causes the server to efficiently iterate through all entries in the configured set of backends, comparing each one against a configured set of data security auditors.

The report is written into a directory structure on the server filesystem. There is a summary.ldif file that provides a summary of all of the processing that was performed. There is also a subdirectory for each of the backends that were examined with a set of LDIF files containing more detailed output from each of those auditors.

Data security auditors included with PingDirectory Server include:

- Identify all entries that contain ACIs.
- Identify administratively disabled users.
- Identify users with passwords that are expired or are about to expire.
- Identify users whose accounts are locked because of too many failed authentication attempts because it's been too long since the user authenticated or because the user did not change their password in a timely manner after an administrative reset.
- Identify users who have multiple passwords.
- Identify users who have explicitly configured privileges or root users or topology administrators who inherit the default set of root privileges.
- Identify users who have passwords encoded with password storage schemes that are considered weak.

The `audit-data-security` tool offers the provided set of command-line arguments in addition to the usual options for connecting and authenticating to the server and scheduling tasks:

`--outputDirectory`

The directory (on the server filesystem) where the report files are created. If this is not specified, then it defaults to a directory whose name reflects the current time in the reports/audit-data-security subdirectory.

--reportFilter

An optional filter that can be used to indicate which entries should be examined. If this is provided, then only entries matching that filter are audited. Otherwise, all entries are examined. This can be provided multiple times if multiple filters should be specified, and only entries matching at least one of those filters are examined.

--backendID

An optional set of backend IDs for the backends to examine. If this is not provided, then all supported backends, including local DB backends and the server configuration, are examined.

--includeAuditor

An optional set of the names for the data security auditors that are invoked. By default, all data security auditors defined in the configuration are included, except for those excluded by the --excludeAuditor argument.

--excludeAuditor

An optional set of the names for the data security auditors that are not invoked.

Soft deletes

Normally, when an authorized client deletes an entry, it is completely removed from the server. However, PingDirectory Server provides support for soft deletes, in which the server preserves the entry instead of removing it. Soft deletes can be used as a means of auditing deleted content or resurrecting content removed either maliciously or in error.

When an entry is soft-deleted, the server makes the following changes to it:

- The server adds the ds-soft-delete-entry auxiliary object class to the entry, which causes it to be hidden from normal search results.
- The server renames the entry to add the entry's entryUUID attribute to the set of RDN attributes. This allows a new entry to be created with the same distinguished name (DN) as the former entry without conflicting with the soft-deleted form of the entry.
- The server adds the following additional operational attributes to the entry:
  - `ds-soft-delete-from-dn` — The original DN for the entry.
  - `ds-soft-delete-timestamp` — The time that the entry was soft-deleted.
  - `ds-soft-delete-requester-dn` — The DN of the user that requested the delete.
  - `ds-soft-delete-requester-ip-address` — The IP address of the client that requested the delete.

Turning deletes into soft deletes provides a way to verify the content of deleted entries. Even though soft-deleted entries are excluded from search results by default, users with the soft-delete-read privilege can retrieve them by including the *soft-deleted entry access request control* in the search request, such as by providing the `--includeSoftDeletedEntries` argument to `ldapsearch`.

There are two ways that regular deletes can be turned into soft deletes. The first is to include the *soft delete request control* in the delete request, such as using the `--softDelete` argument to `ldapmodify`.

However, it is also possible to have the server automatically convert regular deletes into soft deletes. This can be done by creating a soft-delete policy, which can include the following properties:

auto-soft-delete-connection-criteria

Connection criteria that can identify client connections whose deletes should be converted to soft deletes.

auto-soft-delete-request-criteria

Request criteria that can identify delete requests that should be converted to soft deletes.

soft-delete-retention-time

The maximum length of time that soft-deleted entries should be retained in the server.

soft-delete-retain-number-of-entries

The maximum number of soft-deleted entries that should be retained in the server.

After a soft delete policy has been created, the global configuration can be updated to use it with the following property:

soft-delete-policy

The soft-delete policy that should be used by the server. If this is not configured, then soft deletes are disabled in the server, even for delete requests that use the soft delete request control.

Only soft-delete policies that contain values for at least one of the `auto-soft-delete-connection-criteria` and `auto-soft-delete-request-criteria` properties can automatically convert regular deletes to soft deletes. If the server has a soft delete policy without criteria, then soft deletes are only allowed with the soft delete request control.

If necessary, soft-deleted entries can be resurrected with the *undelete request control* in an add request, such as using the `--allowUndelete` argument to `ldapmodify`. The attributes of the add request can be used to provide the details of the undelete. In particular, the DN from the add request specifies the DN to use for the resurrected entry, and the `ds-undelete-from-dn` attribute specifies the DN of the soft-deleted entry to undelete.

Index

A

B

C

D

F

G

H

I

S

T

U

V

W