PingDirectory

July 1, 2025

PINGDIRECTORY
Version: 10.3;latest

Table of Contents

PingDirectory

The PingDirectory suite of products includes the PingDirectory server, the PingDirectoryProxy server, the Delegated Admin application, and the PingDataSync server.

Release Notes

- Current
- Previous releases

Get Started with PingDirectory

PingDirectory

A high-performance, extensible LDAP directory.

- Introduction to the PingDirectory server
- Installing the PingDirectory server
- Upgrade considerations
- Uninstalling the PingDirectory and PingDirectoryProxy servers
- Signing on to and configuring the admin console

PingDirectoryProxy

A fast and scalable LDAPv3 gateway for the PingDirectory server.

- Introduction to the PingDirectoryProxy server
- Installing the PingDirectoryProxy server
- Upgrade considerations
- Uninstalling the PingDirectory and PingDirectoryProxy servers
- Signing on to and configuring the admin console

Delegated Admin

An add-on to PingDirectory that enables the delegation of user and group management.

- Introduction to Delegated Admin
- Installing Delegated Admin
- Upgrade considerations

PingDataSync

An efficient, Java-based server that provides high throughput, low latency, and bidirectional real-time synchronization between two endpoint topologies.

- Introduction to the PingDataSync server
- Installing the PingDataSync server
- Upgrade considerations
- Signing on to and configuring the admin console

Use PingDirectory

- Best Practices: PingDirectory Operational Support
- PingDirectory Server Administration Guide
- PingDirectoryProxy Server Administration Guide
- Delegated Admin Application Guide
- PingDataSync Server Administration Guide

Troubleshoot PingDirectory

- Troubleshooting the PingDirectory and PingDirectoryProxy servers
- Troubleshooting the PingDataSync server

Learn More

- Try PingDirectory
- PingDirectory Community Page
- PingDirectory Customer Training (existing customers only)
- Partner Portal (partners)

Release Notes

Unless otherwise noted, all of the following enhancements, known issues, and resolved issues apply to the PingDirectory server, the PingDirectoryProxy server, and the PingDataSync server.

Subscribe to get automatic updates: PingDirectory Release Notes RSS feed

Notes for 10.3.x versions

PingDirectory suite of products 10.3.0.0 (July 2025)

Removed support for Java 11

Info | DS-49541 | PingDirectory, PingDirectoryProxy, PingDataSync

Support for Java 11 has been removed. You must be running Java 17 or a later supported version, as detailed in the System requirements. Learn more about upgrading a PingDirectory server running Java 11 in Considerations when upgrading to version 10.3.

Support for Internet Explorer 11 has been deprecated

Info | PingDirectory, PingDirectoryProxy, PingDataSync

Support for Internet Explorer 11 has been deprecated and will be removed in a future release.

Support for the `sync-pipe-view` tool has been deprecated

Info | PingDataSync

Support for the `sync-pipe-view` tool has been deprecated, and the tool will be removed in a future release.

Added user entry forwarding for easier request authorization

New | DS-49681 | PingDirectory, PingDirectoryProxy

We added a mechanism to forward the authenticated user's entry to backend servers in an entry-balanced proxy configuration. This change makes it easier to authorize requests in backend sets that don't contain the user's entry.

You can use this mechanism instead of the `authz-dn` property in the entry-balancing request processor configuration or the `ds-authz-map-to-dn` operational attribute in user entries, which are both used to map requests as the authenticated user to a different surrogate user in the other backend sets.

Learn more in Forwarding authorization identities in requests.

Added REST API request controls for soft and hard deletes

New | DS-49530 | PingDirectory

We added support for the following HTTP request controls in the Directory REST API:

Soft delete

Used to soft-delete entries

Hard delete

Overrides automatic soft-delete policies and performs a full hard delete

Soft-deleted entry access

Used to read or search soft-deleted entries

Undelete

Restores soft-deleted entries to their normal state

Learn more in the Directory REST API documentation.

Added support for the HAProxy PROXY protocol

New DS-43335 PingDirectory, PingDirectoryProxy

We added support for LDAP and LDAPS clients accessing the server through a software load balancer using the PROXY protocol, which allows the server to see the actual address and port of the end client system rather than just the address of the load balancer.

The server supports LDAP clients using TCP over IPv4 or IPv6 with header versions 1 and 2. It also accepts valid PROXY protocol headers with other protocols and address families, but it only updates the client address and port for TCP-based clients.

Learn more in Using the HAProxy PROXY protocol.

Added support for a Thales HSM plugin

New DS-48531 PingDirectory

We added support for a Thales HSM plugin, which is available as a separate download. To get the plugin, contact your Ping Identity account representative.

More efficient server handling of failed authentication attempts

Improved DS-49418 PingDirectory

We changed the default server behavior for failed authentication attempts to unavailable user accounts. If a user's account becomes unavailable (for example, because the account is locked, disabled, or the password has expired), the server won't update the user's recent login history for failed authentication attempts.

This change can prevent excessive write operations to a user entry in cases where the user can't possibly authenticate, including when accounts could be subject to password guessing or denial-of-service attacks.

Encoded password caching improved for frequently used passwords

Improved DS-49516 PingDirectory

We improved the eviction logic for encoded password caches to help ensure that frequently used passwords remain cached. When a cache becomes full and needs to add a record, the server evicts the least-recently-used record to make room. Previously, the server evicted the oldest record from the cache.

Smarter `dsreplication initialize` failure behavior

Improved DS-48158 PingDirectory

We improved the **`dsreplication initialize`** failure behavior for source backends supplied in a JSON topology file. A backend must be enabled before it can be initialized. If **`dsreplication initialize`** doesn't successfully initialize the target backend from one source, the command re-enables the target backend before attempting to initialize it from the next source in the JSON file.

Exclude virtual attributes to streamline reversible delete audit logging

Improved DS-49377 PingDirectory

We added a configuration property for the file-based audit log publisher that can exclude virtual attributes from delete audit log records that use the reversible form of logging.

Excluding virtual attributes reduces the size of these log messages and can eliminate the potential performance impact of computing their values. Virtual attributes are still included by default in delete audit log messages generated by the regular file-based audit logger, but they are now suppressed by default in the data recovery log.

Made it easier to update FIPS compliance levels

Improved DS-49550 PingDirectory, PingDirectoryProxy, PingDataSync

We updated **`manage-profile replace-profile`** to allow changing the value of the `--fips-provider` argument in `setup-arguments.txt` from `BCFIPS` to `BCFIPS2`. This makes it possible to update an existing instance running in FIPS 140-2 compliance mode to use FIPS 140-3 compliance mode.

Added FIPS-compliance information to monitor entries

Improved DS-49731 PingDirectory, PingDirectoryProxy, PingDataSync

We updated the Version and SSL Context monitor entries to always include the `fips-compliant-mode`, `fips-140-2-compliant-mode`, and `fips-140-3-compliant-mode` attributes, even when the server is running in non-FIPS-compliant mode. We also exposed those attributes in the **Version** monitor entry in the admin console's **Status** section.

Added on-demand LDAP connection pool creation

Improved DS-49944 PingDirectory, PingDirectoryProxy, PingDataSync

We added an option to allow LDAP external servers to create connection pools without any initial connections so that all connections for use in the pool are created on demand. This can help make it faster to initialize components that use one or more LDAP external servers, but initial attempts to communicate with those servers could take longer as a result of needing to establish new connections.

Clearer attribute parsing for the Processing Time Histogram plugin

Improved DS-49558 PingDirectory, PingDirectoryProxy

We added the `include-parseable-attribute-names` option to the Processing Time Histogram plugin to output entries in a format that's easier to parse. These reformatted entries are duplicates and still exist in their original format. Changing this option requires a server restart to take effect.

Improved server setup when using a profile

Improved | DS-50069 | PingDirectory

We added the `--skipImportLdif` argument to `manage-profile setup`. You can supply this argument to set up a server without importing any LDIF files contained in the profile directory structure.

Fixed a server installation issue with Java 17 and Red Hat

Fixed | DS-49716 | PingDirectory, PingDirectoryProxy, PingDataSync

We fixed an issue that could prevent installing or running servers using Java 17 or later on Red Hat Enterprise Linux (RHEL) systems when the operating system itself is configured to run in FIPS-compliant mode.

This operating system setting is unrelated to whether the PingDirectory server has been set up to run in FIPS-compliant mode.

Fixed replication behavior for `listen-on-all-addresses`

Fixed | DS-49547 | PingDirectory

We fixed the replication server configuration property `listen-on-all-addresses` so that when the property is set to `false`, replication servers only listen to the replication port on the interface that corresponds to the hostname of the server instance for that replication server.

Fixed an issue with replication assurance for some password updates

Fixed | DS-49851 | PingDirectory

We fixed an issue where replication assurance wasn't applied to the internal operation performed by the password modify extended operation.

Fixed an issue with `dsreplication enable`

Fixed | DS-35915 | PingDirectory

We fixed a bug where `dsreplication enable` ignored the `--noPropertiesFile` option and incorrectly applied options from the `tools.property` file.

Fixed the failure behavior for `dsreplication initialize`

Fixed | DS-49890 | PingDirectory

We fixed an issue where a PingDirectory server would continue sending binary data to the destination server after a failed attempt to initialize using `dsreplication initialize`. This behavior interfered with further initialization attempts from any other server.

Fixed a `replace-certificate` trust store issue

Fixed | DS-44645 | PingDirectory, PingDirectoryProxy, PingDataSync

We fixed an issue that prevented the **`replace-certificate`** tool from using the JVM-default trust store when replacing the listener certificate in interactive mode.

Fixed a `replace-certificate` argument issue

Fixed | DS-49769 | PingDirectory, PingDirectoryProxy, PingDataSync

We fixed an issue where **`replace-certificate replace-listener-certificate`** didn't obey the `--trust-store-update-type` argument.

Fixed an issue with some password resets

Fixed | DS-50108 | PingDirectory, PingDirectoryProxy

We fixed an issue where password resets done with the `bypass-pw-policy` privilege would circumvent the `force-change-on-reset` property of password policies.

Fixed an issue with the Modifiable Password Policy State plugin

Fixed | DS-49878 | PingDirectory

We fixed an issue where the Modifiable Password Policy State plugin didn't obey the value of the `filter` property.

Fixed an issue with the Entry Counter plugin

Fixed | DS-49872 | PingDirectory

We fixed an issue where the Entry Counter plugin couldn't evaluate criteria filters against virtual attributes with `require-explicit-request-by-name` set to `true`.

Restored the ability to modify an enabled Entry Counter plugin

Fixed | DS-49816 | PingDirectory

We fixed an issue with the Entry Counter plugin where an enabled plugin couldn't be modified.

Fixed an issue with the Monitor History plugin preserving files

Fixed | DS-46253 | PingDirectory, PingDirectoryProxy, PingDataSync

We fixed an issue where the Monitor History plugin wouldn't preserve files for longer than 14 days when `retain-files-sparsely-by-age` was set to `true`.

Fixed a SCIM issue with modifying `ds-pwp-modifiable-state-json`

Fixed | DS-49781 | PingDirectory

We fixed an issue where SCIM requests that attempted to modify the `ds-pwp-modifiable-state-json` attribute would fail.

Fixed SCIM response errors

Fixed | DS-48511 | PingDirectory, PingDirectoryProxy

We fixed an issue with inconsistencies in `id-attribute` values returned in SCIM operation responses. We also fixed an issue with SCIM GET operations where a filter used to search for an entry would result in a 404 error.

Fixed a SCIM 2.0 PUT issue with attribute values

Fixed | DS-49619 | PingDirectory

We fixed an issue where SCIM 2.0 PUT operations involving multivalued complex attributes would incorrectly remove some of the values.

Fixed a REST API issue with failed PUT requests

Fixed | DS-49912 | PingDirectory

We fixed an issue in the REST API where PUT requests would return a 500 response when attempting to replace the value of a virtual attribute.

Fixed an issue with allowed REST API syntax violations

Fixed | DS-46314 | PingDirectory

We fixed an issue where REST API calls failed due to attribute syntax violations, even though the server had been configured to allow syntax violations for those attributes.

Added a missing debug type to LDAP SDK logging

Fixed | DS-43814 | PingDirectory, PingDirectoryProxy

We added the missing `connection-pool` debug type to the server's support for LDAP SDK debug logging.

Suppressed inaccurate server startup warnings

Fixed | DS-49991 | PingDirectory

We suppressed inaccurate server startup warning messages about some Apache `commons-logging` classes being scanned from multiple locations. The scan detected older versions of those classes packaged inside a Spring JCL `.jar` file needed by the admin console, but the older versions aren't loaded at runtime.

Fixed an internal error logged at server restart

Fixed | DS-49551 | PingDirectory

We fixed a null pointer exception error logged when restarting a PingDirectory server configured with Delegated Admin.

Fixed a Delegated Admin memory leak

Fixed DS-49409 PingDirectory

We fixed an issue where Delegated Admin could leak memory due to unfinalized memory consumers.

Fixed a topology issue related to removing defunct servers

Fixed DS-49700 PingDataSync

We fixed an issue where the `remove-defunct-server` tool could leave a PingDataSync topology in a state where new servers couldn't be added.

Fixed an issue with password sync from Active Directory

Fixed DS-50043 PingDataSync

We fixed an issue where password synchronization from multiple Active Directory subdomains through multiple sync pipes could fail abruptly.

Excluded some password attributes from sync sources

Fixed DS-49212 PingDataSync

We changed the `resync` tool to exclude `unicodePwd` automatically from AD sync sources and `password` from PingOne sync sources.

By design, the `resync` tool updates the existing values for included attributes at the destination to match what's found at the source. If `resync` can't retrieve an attribute value at the source, it removes any existing values at the destination. Because `resync` can't retrieve these password attributes from their sources, we've excluded them from the attributes for `resync` consideration to avoid disrupting the values at the destination.

You can still include these attributes manually in a `resync` operation by providing the `--includeSourceAttr` argument.

Fixed an issue with logging changes to some attributes

Fixed DS-49917 PingDataSync

We fixed an issue where PingDataSync would log an operation as not applied if the only changes applied were to password policy state attributes.

Fixed an issue with third-party change detectors

Fixed DS-49035 PingDataSync

We fixed an issue where third-party change detectors didn't properly persist the state of processing at the sync source. This could've caused change detector malfunctions with the `set-startpoint` task, with saving the change detector's state upon server shutdown, or with communicating that state to failover instances.

Fixed an issue with the SDK LDAP sync destination plugins

Fixed DS-49854 PingDataSync

We fixed an issue in the server SDK's example LDAP sync destination plugins, where the plugins dropped modify DN operations that didn't affect the RDN.

Notes for previous 10.x versions

PingDirectory suite of products 10.2.0.1 (March 2025)

Fixed an issue with FIPS compliance and Oracle JDK 17+

Fixed DS-48832 PingDirectory, PingDirectoryProxy, PingDataSync

We've fixed an issue where the server wouldn't install or operate correctly in FIPS-compliant mode with Oracle JDK 17 or later.

Fixed an internal error logged at server restart

Fixed DS-49551 PingDirectory

We've fixed a null pointer exception error logged when restarting a PingDirectory server configured with Delegated Admin.

PingDirectory suite of products 10.2.0.0 (December 2024)

Basic authentication notice for the REST API

Security PingDirectory, PingDirectoryProxy

Basic authentication has been deprecated.

Although basic authentication is currently enabled by default for the PingDirectory REST API, it could be disabled by default in a future release. Learn more about enabling or disabling basic authentication in Directory REST API configuration.

Fixed a potential basic authentication vulnerability for the REST API

Security DS-49261 PingDirectory, PingDirectoryProxy

Fixed a potential user enumeration vulnerability when using the Directory Rest API with basic authentication.

SSO sessions to the admin console now expire with the access token

Security DS-49005, DS-49006, DS-49009 PingDirectory

When you enable single sign-on (SSO) for the PingDirectory admin console, the session expiration now depends on the lifetime of the OIDC access token. This change prevents the following:

- Access to server information with an expired token

- Server crashing caused by an increasing number of open connections created when attempting to bind to the server with an expired token
- Unresponsive `Connection Error` pages when switching between servers in a topology

Support for Java 11 has been deprecated

Info PingDirectory, PingDirectoryProxy, PingDataSync

Support for Java 11 has been deprecated and will be removed in a future release.

Support for SCIM 1.1 has been deprecated

Info PingDirectory, PingDirectoryProxy

Support for SCIM 1.1 has been deprecated and will be removed in a future release.

SNMP is deprecated

Info PingDirectory, PingDirectoryProxy, PingDataSync

SNMP is deprecated.

Some Java properties set automatically for Java 21 support

Info DS-49414 PingDirectory, PingDirectoryProxy, PingDataSync

To help enable runtime support of Java 21, the **`import-ldif`** and **`rebuild-index`** tools now automatically set the `java.security.manager` property to `allow` in the following places that affect those tools:

- The following properties in `config/java.properties`:
  - `start-server.java-args`
  - `import-ldif.offline.java-args`
  - `rebuild-index.offline.java-args`
- Programatically, when **`setup`** invokes **`import-ldif`**

  This property change allows these tools to use the backend database `DbCacheSize` utility by preventing that utility from exiting.

Custom SDK extensions using Javax will need to be migrated and recompiled in 10.3

Info PingDirectory, PingDirectoryProxy, PingDataSync

Several components will be upgraded in version 10.3 of the PingDirectory suite of products. If any of your custom Server SDK extensions have classes that import `javax.*` packages, you will need to migrate them to the equivalent `jakarta.*` packages and then recompile the extensions.

SCIM 2 SDK version compatibility for custom extensions

Info PingDirectory, PingDirectoryProxy

For custom SCIM 2.0 extensions that use the UnboundID SCIM 2 SDK with objects under the Javax namespace (for example, `javax.ws.rs.client.WebTarget` or `javax.ws.rs.core.MediaType`), you must use version 2.4.0 of the SDK.

Starting with version 3.0.0, the SDK uses objects under the Jakarta namespace (such as `jakarta.ws.rs.client.WebTarget`), which aren't compatible with PingDirectory 10.2.0.0 or earlier.

> **Tip**
>
> If your SCIM 2.0 extension doesn't use objects with Javax namespaces, you can use later versions of the UnboundID SCIM 2 SDK.

Added runtime support for Java 21

New DS-48833, DS-49193 PingDirectory, PingDirectoryProxy, PingDataSync

Added JRE support for Oracle JDK 21 and OpenJDK 21.

Added support for Generational ZGC garbage collection

New DS-49408 PingDirectory, PingDirectoryProxy, PingDataSync

Added support for Generational ZGC garbage collection on servers running Java 21. Learn more in JVM garbage collection using ZGC.

Added support for FIPS 140-3

New DS-49249, DS-49285 PingDirectory, PingDirectoryProxy, PingDataSync

Added support for setting up the server in FIPS 140-3-compliant mode using 2.x versions of the Bouncy Castle FIPS-compliant library.

To set up the server in FIPS 140-3-compliant mode, use the `--fips-provider BCFIPS2` argument. You can still set up the server in FIPS 140-2-compliant mode using the `--fips-provider BCFIPS1` argument.

Learn more in Setting up the server in FIPS-compliant mode.

Use OAuth scopes for ACIs on REST API endpoints

New DS-48851 PingDirectory, PingDirectoryProxy

To help isolate access to admin credentials in authentication workflows, we added the ability to use OAuth scopes to enforce ACIs for users authenticating to most Directory REST API endpoints.

When users send authentication requests with an OAuth 2.0 bearer token, they can be granted OAuth scopes by a token validator, such as PingFederate. Scope-configured PingDirectory ACIs can then be applied to those scopes, providing the permissions for the user and the request.

Learn more in Using OAuth scopes for ACI rules with the REST API.

Keep count of specific entries with a new plugin

New DS-422, DS-47690 PingDirectory

Added an entry counter plugin that can determine the number of entries in the server matching configured sets of base DN and filter criteria. The plugin returns the resulting entry counts in monitor entries and can optionally include information about the amount of space used to store those entries in the backend database.

You can also define warning and error threshold values for each criteria. If the number of matching entries reaches those thresholds, the server raises a warning or error alarm.

Learn more in Working with the entry counter plugin.

Monitor the risk of performance degradation

New DS-49116 PingDirectory

Added the `db-on-disk-to-db-cache-size-ratio` monitor attribute to database environment monitor entries. Also added a gauge to monitor the attribute and raise an alert if the on-disk database size becomes eight times larger than the size of the in-memory cache, which could cause an increased risk of performance degradation.

Learn more about this monitor attribute in Server gauges.

Added key and trust manager caching

New DS-49135 PingDirectory, PingDirectoryProxy, PingDataSync

Added the ability to cache key managers and trust managers to prevent loading keystore and truststore files from disk when establishing connections to process requests. Use the `enable-key-manager-caching` and `enable-trust-manager-caching` configuration properties to enable or disable caching.

Learn more about key and trust manager caching in Configuring key and trust manager providers.

Added caching for expensive password storage schemes

New DS-49100 PingDirectory, PingDirectoryProxy, PingDataSync

Added a secure, in-memory cache for improving the performance of repeated authentication attempts for users with passwords encoded using expensive storage schemes, including PBKDF2, Argon2, bccrypt, and scrypt.

When a user whose password is encoded with one of these schemes tries to authenticate, the server checks the cache to see if it contains their encoded password. If not, the server verifies the password using the expensive processing required by the storage scheme and then adds it to the cache, along with a salted SHA-256-encoded representation of that password. On later authentication attempts, if the cache includes the expensive encoded password, the server can use the salted SHA-256-encoded variant to verify the password much more quickly.

The cache only holds encoded representations of passwords using the expensive storage scheme and the faster salted SHA-256 digest. It doesn't include the plaintext representations of passwords or information that could associate an encoded password with the corresponding user account. The contents of the cache aren't written to disk or persisted in any other way.

You can adjust the cache size by using the `encoded-password-cache-size` property in the password storage scheme configuration. Setting this property to `0` disables caching for that scheme.

Learn more in Encoded password caching.

Better authentication performance for dynamic groups

Improved | DS-49030 | PingDirectory, PingDirectoryProxy

Dramatically reduced the time needed to authenticate in environments with a very large number of dynamic groups.

No server lockdown for missing changes from obsolete replicas

Improved | DS-49070 | PingDirectory

Changed replication behavior to prevent server lockdown for missing changes due to obsolete replicas. This change affects the following scenarios where, previously, these types of missing changes triggered lockdowns:

- The `replication-purge-obsolete-replicas` global configuration property is set to `false`.
- Not all servers in the topology support configurable missing changes.
- The remote server indicates lockdown for replicas that are actually obsolete.

No server lockdown for missing changes after a restart

Improved | DS-48972 | PingDirectory

Changed the default missing changes policy from `favor-integrity` to `favor-availability` to prevent lockdowns for missing changes from persisting between server restarts.

Made it easier to repair topologies with missing changes

Improved | DS-49063 | PingDirectory

Updated the `check-replication-domains` tool to distinguish between deleted and obsolete replicas, making it easier to manage missed changes when repairing a topology.

A replica listed as `DELETED` has been deleted from the topology but isn't yet obsolete. A replica listed as `OBSOLETE` has been deleted from the topology and only contains changes older than the replication purge delay.

Mitigated a potential slowdown to server backups

Improved | DS-49227 | PingDirectory

Updated the server to attempt to pause backend cleaning activity while backups are in progress. If a cleaner thread removes any database files during a backup, it can cause the server to include additional files in the backup, which increases both the size of the backup and the time required to generate it. This effect can intensify when performing a rate-limited backup.

Improved password sync functionality

Improved | DS-48794 | PingDataSync

Added the ability to sync password changes and modifiable password policy state changes at the same time to PingDirectory sync destinations.

Allowed proxied requests for HTTP external servers

Improved | DS-48729 | PingDirectory, PingDirectoryProxy, PingDataSync

Updated the HTTP external server configuration to allow requests to be forwarded through an HTTP proxy server.

Made it easier to change garbage collection types

Improved | DS-48966 | PingDirectory, PingDirectoryProxy, PingDataSync

Added the `--gcType` argument for the **`dsjavaproperties`** tool to make it easier to change the server garbage collection type. Depending on the platform and Java version, some garbage collection types might not be recommended or supported.

Learn more in Changing the JVM garbage collector type.

Better compatibility with security scanners

Improved | DS-48850 | PingDirectory

To improve compatibility with third-party security scanners, HTTP response headers specified in the HTTP Connection Handler `response-header` property are now included in all error responses, for example `404 NOT FOUND`.

Reduced redundant logging

Improved | DS-49124 | PingDirectory

Lowered the default global configuration property `duplicate-error-log-limit` from 2000 to 200 to reduce redundant logging.

Removed the restart prompt when changing a certificate alias

Fixed | DS-45174 | PingDirectory, PingDirectoryProxy, PingDataSync

Removed the prompt to restart the LDAP connection handler component after changing the `ssl-cert-nickname` configuration property because a restart isn't required.

Fixed an issue with parsing the `JAVA_HOME` path

Fixed | DS-DS-49349 | PingDirectory, PingDirectoryProxy, PingDataSync

Fixed an issue where the **`dsjavaproperties`** tool didn't correctly parse the `JAVA_HOME` path and threw the following error:

```
The old java.properties JAVA_HOME was changed because the Java installation in
it is not valid. This is usually because the previous JAVA_HOME pointed to an
incompatible version of Java. The previous JAVA_HOME value will
remain in the generated java.properties.old file.
```

Changed the `collect-support-data` monitor file behavior

Fixed | DS-47384 | PingDirectory, PingDirectoryProxy, PingDataSync

Changed the `collect-support-data` tool to use the latest `monitor-history` file if it can't find `ldap/monitor.ldif` when examining monitor data.

Fixed an issue with VLV index errors

Fixed | DS-49296 | PingDirectory, PingDirectoryProxy, PingDataSync

Fixed an issue where untrusted VLV indexes would throw errors for irrelevant searches.

Fixed an issue with Prometheus HTTP servlet error messages

Fixed | DS-49161 | PingDirectory, PingDirectoryProxy, PingDataSync

Fixed an issue where the Prometheus HTTP servlet would publish an excessive number of error messages to the error log when it lost connection to its remote counterpart.

Fixed an issue with `config-diff`

Fixed | DS-49071 | PingDirectory, PingDirectoryProxy, PingDataSync

Fixed an issue where running the **`config-diff`** tool would result in an `Unknown property` error when comparing configuration objects of different types.

Removed suppression messages for disabled alerts

Fixed | DS-49119 | PingDirectory, PingDirectoryProxy, PingDataSync

Fixed an issue where alerts types that were disabled would still output suppression messages.

Fixed server startup with third-party key manager providers

Fixed | DS-49121 | PingDirectory, PingDirectoryProxy, PingDataSync

Fixed an issue that could prevent the server from starting when configured to use a third-party key manager provider created using the Server SDK.

Fixed an issue with the Dictionary Password Validator

Fixed | DS-47914 | PingDirectory, PingDirectoryProxy, PingDataSync

Fixed an issue with the default value for `dictionary-file` in the Dictionary Password Validator configuration.

Supplied missing replication error information

Fixed | DS-48785 | PingDirectory, PingDirectoryProxy, PingDataSync

Fixed an issue where **`dsreplication enable`** didn't print error information if the tool failed to establish a connection to a source or target server.

Fixed failing server upgrades using server profiles

Fixed DS-49374 PingDirectory

Fixed an issue where server upgrades using `manage-profile replace-profile` failed because the default topology admin user account was reverting to the default password policy.

Fixed an issue with updating `isMemberOf` values

Fixed DS-49371 PingDirectory

Fixed an issue where the server didn't properly update `isMemberOf` values as a result of a modify DN operation that renamed or moved a subtree containing one or more groups. The `isMemberOf` values for those groups would continue to reflect their former DNs until a server restart.

Fixed an issue with `ds-backend-entry-count`

Fixed DS-49394 PingDirectory

The `ds-backend-entry-count` monitor attribute for the `replicationChanges` backend now correctly handles sequence number rollover so that it is accurate when more than 2,147,483,647 changes have been made.

Fixed an issue with rebuilding substring indexes

Fixed DS-49384 PingDirectory

Fixed an issue where rebuilding a substring index could either result in an error or the index not properly indexing all keys.

Removed an irrelevant validator error

Fixed DS-42343 PingDirectory

Removed an irrelevant validator error. This error was previously raised on shutdown if a server was concurrently receiving mirror subtree data from the master server.

Fixed an HTTP Connection Handler error message

Fixed DS-49364 PingDirectory

Fixed an error message caused by an improperly configured HTTP Connection Handler object so that it properly lists out the affected object names.

Fixed a `userRoot` issue for server upgrades

Fixed DS-49281 PingDirectory

Fixed an issue that caused some server upgrades to fail. During an upgrade, the update tool added `userRoot` entries for inverted static group support to the server configuration after the `userRoot` backend had been removed.

Fixed an issue with JSON-object extensible matching filters

Fixed | DS-45519 | PingDirectory

Fixed an issue where JSON-object extensible matching filters of type less-than were being evaluated as type less-than-or-equals.

Fixed an issue with lost replication changes and gateways

Fixed | DS-45976 | PingDirectory

Fixed an issue where replication changes could be lost when sent to a location whose gateway was starting or stopping.

Fixed an issue with composite index metadata

Fixed | DS-48969 | PingDirectory

Fixed an issue that could cause an inconsistency in the metadata for a composite index record, which could have the following effects:

- Validator error messages show up in the server's error log.
- The server returns errors in response to some requests.
- Under rare circumstances, the server can't bring the affected backend online.

In addition, the server now has added resiliency against these kinds of issues, allowing it to better identify the correct result set and more effectively notify administrators if it does happen.

Fixed an issue with authorization IDs and the REST API

Fixed | DS-49195 | PingDirectoryProxy

Fixed an issue where the Directory REST API didn't use alternate authorization identities in entry-balanced proxy environments.

Changed proxy transformation requirements for mapped attributes

Fixed | DS-48958 | PingDirectoryProxy

Updated the attribute mapping proxy transformation to require that both the source and target attribute types are defined in the local schema.

This change ensures that the server uses the correct logic when interacting with values of those attributes (for example, to identify whether the attribute type is declared as single-valued or multivalued so that it can properly format the values in REST API responses). The server now prevents adding a new instance of this proxy transformation if either of the attribute types isn't defined in the schema. It also logs a warning message on startup if any existing instance of the transformation references an undefined attribute type.

Fixed an issue with character encoding for PingOne sync destinations

Fixed | DS-49362 | PingDataSync

Fixed an issue where an empty space character didn't get properly encoded when URLs were sent to PingOne sync destinations.

Fixed an issue with filtering virtual attributes

Fixed | DS-49290 | PingDataSync

Fixed an issue with defining which entries to sync with the `include-filter` property in a sync class. If the filter targeted virtual attributes with a `!` (not) operator, the sync operation would fail to exclude matching entries.

Fixed an issue with error messages for sync operations

Fixed | DS-49224 | PingDataSync

Fixed an issue where error messages related to creating sync operations were being logged to all sync pipe log publishers rather than just the associated log publisher.

Fixed an issue with Kafka and `OutOfMemory` errors

Fixed | DS-48762 | PingDataSync

Fixed an issue where failover instances configured with `KafkaSyncDestinations` could leak `KafkaProducer` objects and eventually encounter `OutOfMemory` errors.

Fixed a potential NPE when loading a sync source

Fixed | DS-49248 | PingDataSync

Fixed a potential `NullPointerException` that could occur when attempting to load changes from a sync source.

Expired certificates in keystores after running `replace-certificate`

Issue | DS-49269 | PingDirectory, PingDirectoryProxy, PingDataSync

When replacing a PingDirectoryProxy or PingDataSync certificate with the **`replace-certificate`** tool, the old certificate gets stored in the existing keystore with an alias of `.old`. Instances of PingDirectory in the configuration don't identify this alias as an indicator of a deprecated certificate and can select the `.old` certificate for use in TLS communication, potentially causing a communication failure.

To avoid this issue, you can do one of the following:

1. Set a `Null` key manager provider for all external PingDirectory servers in the configuration.
2. Use the **`manage-certificates delete-certificate`** command to remove unused aliases from the PingDirectoryProxy or PingDataSync keystores.

Learn more about the solutions for this issue in Communication failure due to aliased expired certificate (requires authentication).

Installing in FIPS-compliant mode with Oracle JDK 17 or later

Issue | DS-48832 | PingDirectory

If you attempt to install PingDirectory in FIPS-compliant mode while running Oracle JDK 17 or later, the installation might fail with an error similar to the following:

```
Initializing ..... An error occurred while attempting to initialize the crypto
manager: due to an exception in the Java security provider:
NoSuchAlgorithmException: 1.2.840.113549.1.1.4 Signature not available
```

To avoid this issue, use OpenJDK versions 17 or 21.

Support for the HashiCorp Vault secrets engine

Issue DS-49305 PingDirectory

Currently, the PingDirectory server only supports version 1 of the HashiCorp Vault KV secrets engine. Learn more about KV version 1 in the Vault KV secrets engine documentation.

PingDirectory suite of products 10.1.0.3 (May 2025)

Added key and trust manager caching

New DS-49135 PingDirectory, PingDirectoryProxy, PingDataSync

We've added the ability to cache key managers and trust managers to prevent loading keystore and truststore files from disk when establishing connections to process requests. Use the `enable-key-manager-caching` and `enable-trust-manager-caching` configuration properties to enable or disable caching.

Learn more about key and trust manager caching in Configuring key and trust manager providers.

Fixed a server installation issue with Java 17 and Red Hat

Fixed DS-49716 PingDirectory, PingDirectoryProxy, PingDataSync

We've fixed an issue that could prevent installing or running servers using Java 17 or later on Red Hat Enterprise Linux (RHEL) systems when the operating system itself is configured to run in FIPS-compliant mode.

This operating system setting is unrelated to whether the PingDirectory server has been set up to run in FIPS-compliant mode.

Fixed an issue with FIPS compliance and Oracle JDK 17+

Fixed DS-48832 PingDirectory, PingDirectoryProxy, PingDataSync

We've fixed an issue where the server wouldn't install or operate correctly in FIPS-compliant mode with Oracle JDK 17 or later.

Fixed a `userRoot` issue for server upgrades

Fixed DS-49281 PingDirectory

We've fixed an issue that caused some server upgrades to fail. During an upgrade, the update tool added `userRoot` entries for inverted static group support to the server configuration after the `userRoot` backend had been removed.

Fixed a server startup issue

Fixed DS-49121 PingDirectory, PingDirectoryProxy, PingDataSync

We've fixed an issue that could prevent the server from starting when configured to use a third-party key manager provider created using the Server SDK.

Removed the restart prompt when changing a certificate alias

Fixed DS-45174 PingDirectory, PingDirectoryProxy, PingDataSync

We've removed the prompt to restart the LDAP connection handler component after changing the `ssl-cert-nickname` configuration property because a restart isn't required.

Fixed an internal error logged at server restart

Fixed DS-49551 PingDirectory

We've fixed a null pointer exception error logged when restarting a PingDirectory server configured with Delegated Admin.

Fixed SCIM response errors

Fixed DS-48511 PingDirectory, PingDirectoryProxy

We've fixed an issue with inconsistencies in `id-attribute` values returned in SCIM operation responses. We've also fixed an issue with SCIM `GET` operations, where a filter used to search for an entry would result in a 404 error.

Fixed a SCIM 2.0 `PUT` issue with attribute values

Fixed DS-49619 PingDirectory

We've fixed an issue where SCIM 2.0 `PUT` operations involving multivalued complex attributes would incorrectly remove some of the values.

Fixed a SCIM issue with modifying `ds-pwp-modifiable-state-json`

Fixed DS-49781 PingDirectory

We've fixed an issue where SCIM requests that attempted to modify the `ds-pwp-modifiable-state-json` attribute would fail.

Changed proxy transformation requirements for mapped attributes

Fixed DS-48958 PingDirectoryProxy

We've updated the attribute mapping proxy transformation to require that both the source and target attribute types are defined in the local schema.

This change ensures that the server uses the correct logic when interacting with values of those attributes (for example, to identify whether the attribute type is declared as single-valued or multivalued so that it can properly format the values in REST API responses). The server now prevents adding a new instance of this proxy transformation if either of the attribute types isn't defined in the schema. It also logs a warning message on startup if any existing instance of the transformation references an undefined attribute type.

Removed suppression messages for disabled alerts

Fixed | DS-49119 | PingDirectory, PingDirectoryProxy, PingDataSync

We've fixed an issue where alert types that were disabled would still output suppression messages.

Fixed an issue with authorization IDs and the REST API

Fixed | DS-49195 | PingDirectoryProxy

We've fixed an issue where the Directory REST API didn't use alternate authorization identities in entry-balanced proxy environments.

Fixed an issue with character encoding for PingOne sync destinations

Fixed | DS-49362 | PingDataSync

We've fixed an issue where an empty space character didn't get properly encoded when URLs were sent to PingOne sync destinations.

PingDirectory suite of products 10.1.0.2 (September 2024)

Changed replication to prevent lockdown for missing changes from obsolete replicas

Improved | DS-49070 | PingDirectory

Changed replication behavior to prevent server lockdown for missing changes due to obsolete replicas. This change affects the following scenarios where, previously, these types of missing changes triggered lockdowns:

- The `replication-purge-obsolete-replicas` global configuration property is set to false.
- Not all servers in the topology support configurable missing changes.
- The remote server indicates lockdown for replicas that are actually obsolete.

Made it easier to upgrade replicated servers to version 10.1.0.2 or later

Improved | DS-48798, DS-49090 | PingDirectory

When upgrading a pre-9.2 PingDirectory server in a replicated topology to version 10.1.0.2 or later, the **`update`** tool will automatically set `replication-purge-obsolete-replicas` to false for that server, if not already explicitly configured.

This change helps avoid unintended consequences when upgrading a pre-9.2 replicated server, as the `replication-purge-obsolete-replicas` configuration property has a value of true by default in version 9.2.

After upgrade, the update tool also displays a message with more information:

```
In the 9.2.0.0 release, the implicit default value for the 'replication-purge-obsolete-replicas'
global configuration property changed from 'false' to 'true'. However, it should generally only
be set to true if all servers in the topology are at version 9.2.0.0 or later. Because this server
is being updated from a pre-9.2.0.0 version, it is possible that there are still other pre-9.2.0.0
servers in the topology. As such, the 'replication-purge-obsolete-replicas' property will be explicitly
set to false for this server if it was not explicitly set. Once you have completed the upgrade across all
servers in the topology so that there are no more pre-9.2.0.0 replicas, consider manually setting this
property to 'true' on all servers.
```

Fixed a missing replication error message

Fixed | DS-48785 | PingDirectory, PingDirectoryProxy, PingDataSync

Fixed an issue where `dsreplication enable` wouldn't print error information if the tool failed to establish a connection to a source or target server.

Fixed a `config-diff` error

Fixed | DS-49071 | PingDirectory, PingDirectoryProxy, PingDataSync

Fixed an issue where `config-diff` would result in an `Unknown property` error when comparing configuration objects of different types.

PingDirectory suite of products 10.1.0.0 (June 2024)

Fixed a PingDirectoryProxy authentication issue

Security | DS-48028 | PingDirectoryProxy

Fixed an issue that could have allowed clients attempting to authenticate through the PingDirectoryProxy server to obtain more information in the bind response than would have been allowed if the request had been sent directly to a PingDirectory server.

Added presence component support for composite index filter patterns

New | DS-18120 | PingDirectory

Added the ability to use presence components in composite indexes, whether as a standalone filter pattern or in an AND filter pattern. You can now replace existing presence attribute indexes with composite indexes for improved scalability or to limit the scope of index keys by using a base DN pattern. Learn more about Composite index filter patterns.

Added static equality support for composite index filter patterns

New | DS-18120 | PingDirectory

Added the ability to use equality components with static values in composite index filter patterns, which can be useful in cases where you want to index specific attribute values that are present in a large number of entries. The index filter pattern can either be a simple static equality component, an AND filter with multiple static equality components, or an AND filter with static equality components combined with other supported filter pattern components.

Added approximate matching support for composite index filter patterns

New DS-48631 PingDirectory

Added the ability to use approximate matching components in composite indexes, whether as a standalone filter pattern or in an AND filter pattern. You can now replace existing approximate matching attribute indexes with composite indexes for improved scalability or to limit the scope of index keys by using a base DN pattern.

Added support for localized matching in searches

New DS-48630 PingDirectory

Added support for several collation matching rules, which allow clients to use extensible match filters to better search for entries with non-English values. Learn more about Localization of searches with collation matching.

Added a repair tool for broken trust in replicated topologies

New DS-48752 PingDirectory

Added a tool to repair broken listener certificate trust in replicated topologies. To reduce troubleshooting and speed up the repair of broken subtree mirroring in a replicated topology where listener certificates have fallen out of trust, you can use the `repair-topology-listener-certificates` tool. Learn more about Repairing broken listener certificate trust in replication.

> **Note**
>
> This tool isn't an alternative to using the `replace-certificates` tool when changing listener certificates normally and can only be used to address issues that arise from unsuccessful certificate updates in the topology registry.

Added the ability to compare LDAP schemas between servers

New DS-47930 PingDirectory, PingDirectoryProxy

Added the `compare-ldap-schemas` tool to identify differences between the schemas of two LDAP servers.

Added a configurable limit for subtree modification

New DS-47316 PingDirectory

Added the `subtree-modify-dn-size-limit` configuration property for local DB backends. By default, the server now rejects modify DN operations in which the target entry has more than 100 subordinate entries, which can help protect against inadvertent and potentially expensive subtree moves or renames.

With this property, subtree modify DN operations can be completely disabled, limited to subtrees of a specified maximum size, or allowed for subtrees of any size.

Added client connection info in request-type access logging

New DS-48614 PingDirectory

Added the `include-connection-details-in-request-messages` property to allow you to add details about client connections in request-type access log messages. The property is disabled by default. Learn more about Adding connection information to request-type log messages.

Added the ability to exclude error log messages

New DS-48581 PingDirectory, PingDirectoryProxy, PingDataSync

Added the ability to exclude specific error log messages to help simplify server administration. You can configure several criteria to determine which messages to exclude. Learn more about Excluding specific log messages.

Added boolean attribute support for Prometheus metrics

New DS-47286 PingDirectory

Added support for boolean attributes in Prometheus monitor metrics. These metrics can be used for monitor attributes that have values such as `true`, `false`, `enabled`, `disabled`, `yes`, `no`, `on`, `off`, `1`, or `0`. The server sends a gauge metric to Prometheus with a value of `1` or `0` to represent these values. Learn more about Customizing published metrics.

Added obfuscation for sensitive Kafka values

New DS-48216 PingDataSync

Added the `sensitive-kafka-producer-property` configuration object to enable you to obscure sensitive producer property values, such as keys or passwords. Learn more about Obscuring sensitive producer property values.

Added support for PKCS11 key wrapping transformations

New DS-48514 PingDirectory

For environments that require specific key wrapping transformations, we added the ability to use `dsconfig` to update the `key-wrapping-transformation` property for PingDirectory PKCS11 cipher stream providers.

Added a password verification extended operation

New DS-48662 PingDirectory, PingDirectoryProxy

Added support for an extended operation to verify passwords, which can be used to determine whether a specified password is correct for a given user without performing any other password policy processing. Support for this operation is disabled by default. Learn more about The verify password extended operation.

Added support for synchronizing account lock statuses from PingOne

Improved DS-47933 PingDataSync

Increased the consistency of enterprise-wide user statuses by adding support for synchronizing account lock status events from a PingOne source. Learn more about Synchronizing PingOne account status with PingDirectory.

Enabled candidate set caching to improve indexed search performance

Improved DS-48530 PingDirectory

Added a configuration property that enables you to cache the candidate set for indexed search requests that include the simple paged results request control. By default, the server recomputes the candidate set for each page of results retrieved from the server. With caching enabled, the server can reuse the same candidate set across all pages without needing to recompute it each time.

Learn more about optimizing paged searches using caching.

Reduced the performance impact of exploded index cleanup processing

Improved DS-48672 PingDirectory

Reduced the performance impact of the background cleanup processing that occurs when an exploded index key exceeds the index entry limit.

Previously, performance of other write operations had been substantially degraded while the cleanup was in progress and, under certain circumstances, could have caused the server to appear unresponsive. Now, the background cleanup processing might take significantly longer but has much less impact on other operations while that cleanup is in progress.

Increased the speed of search results

Improved DS-48075 PingDirectory

Updated the server to allow it to start returning matching entries more quickly and with reduced memory consumption when processing a search request that can be perfectly satisfied by a single composite index key.

Increased the server startup speed

Improved DS-48869 PingDirectory, PingDirectoryProxy, PingDataSync

Changed the default behavior of the interactive setup to not prime the database by preloading its contents.

Increased throughput in backend DB environments

Improved DS-48827 PingDirectory

Increased write throughput and significantly reduced response time outliers in backend DB environments.

Improved performance for servers with large configuration archives

Improved DS-48875 PingDirectory, PingDirectoryProxy, PingDataSync

Changed the configuration archive to retain a maximum of 100 previous configurations by default to alleviate the performance impact of large archives.

Improved server guidance around attribute and composite indexes

Improved DS-48670, DS-5357 PingDirectory

Updated the server to raise an alert or log a warning message when attribute index entry limits are set too high and to recommend the use of composite indexes instead. High index entry limits can lead to performance issues for attribute indexes, and composite indexes offer much better performance and scalability for index keys that match a large number of entries.

Reduced memory pressure for dynamic group caching

Improved DS-44929 PingDirectory

Reduced the amount of memory needed to cache information about dynamic groups.

Enabled data imports to ignore duplicate attribute values

Improved DS-48603 PingDirectory

Updated the `import-ldif` tool to add an `--ignoreDuplicateAttributeValues` argument. By default, the tool rejects any entries that contain duplicate values within the same attribute, but this new argument causes it to behave as if each value had only been provided once.

Enhanced the configurability of ACI rights for adding entries

Improved DS-48516 PingDirectory

Added the `evaluate-target-attribute-rights-for-add-operations` configuration property to the access control handler to correct a behavior where the bind user required an `allow add` ACI for only one attribute of an entry to add the entry.

With this property enabled, the bind user must have an `allow add` ACI for all attributes of an entry to add the entry. To avoid changing existing functionality, `evaluate-target-attribute-rights-for-add-operations` is disabled by default. Learn more about Changing the allow add ACI behavior for entries.

Increased replication speed

Improved DS-48826 PingDirectory

Increased throughput for replicated operations.

Made schema replication more efficient

Improved DS-48343 PingDirectory

Made schema replication more efficient by not sending, and by not applying, update messages that don't need to be applied. This is done by calculating the generation ID correctly, setting replication operational attributes in the schema backend, and by noting the changes most recently applied in the `replicationChanges` backend.

Improved obsolete replica logic

Improved DS-48800 PingDirectory

Improve obsolete replica logic so that replication more accurately determines if a replica is obsolete.

Increased the efficiency of replication backlog health checks

Improved DS-48552 PingDirectoryProxy

Made the server health check for the replication backlog more efficient.

Reduced the size of replication monitor messages

Improved DS-48058 PingDirectory

To reduce the size of replication monitor messages, the `include-all-remote-servers-state-in-monitor-message` global configuration property is now set to false by default. Servers no longer include information about other remote servers in their monitor messages, but each server describes itself with its own monitor message.

Reduced the retrieval time for the percentage of undeletable files

Improved DS-45172 PingDirectory

Used caching to speed up the Database Environment monitor entry retrieval of the percentage of undeletable database files.

Expanded the controls for `export-reversible-passwords`

Improved DS-48022 PingDirectory

Updated the **`export-reversible-passwords`** tool to allow you to specify base DNs for entries to include in or exclude from the export.

Made it easier to upgrade the Password Sync Agent

Improved DS-17945, DS-48793 PingDataSync

Made it easier to install and upgrade the Password Sync Agent by clarifying and expanding the documentation.

Enhanced debug support for CLI tools

Improved DS-48239 PingDirectory, PingDirectoryProxy, PingDataSync

Added debug logging support to a number of command-line tools. Use the `--help-debug` argument to see the relevant arguments.

Added a timeout for long-running exec alert commands

Improved DS-48724 PingDirectory

Added a timeout feature that automatically terminates the execution of a long-running command or script initiated by the exec alert handler. The `command-timeout` attribute controls the time limit and has a default value of 1 hour. To disable this timeout, you can change the `command-timeout` value to `0 s`. Learn more about Changing the timeout for an exec alert handler.

Enabled expensive operations access logging by default

Improved DS-48856 PingDirectory, PingDirectoryProxy, PingDataSync

Made a configuration change to have the expensive operations access logger enabled by default. Any operations that take at least one second to complete will be logged to the `logs/expensive-ops` file.

Added cipher re-initialization logic for performance improvement

Improved | DS-48893 | PingDirectory

Added the `always-reinitialize-cached-cipher-instances` configuration property to specify whether ciphers retrieved from an internal cache should always be re-initialized using `Cipher.init()` before re-use, or whether re-initialization can be skipped if the cipher hasn't been used to encrypt or decrypt data since a previous call to `Cipher.init()` or `Cipher.doFinal()`.

This new property defaults to `true`, unless the server is running in FIPS 140-2-compliant mode. Skipping unnecessary re-initialization of cached ciphers results in greatly improved performance for implementations such as BCFIPS AES/CBC/PKCs5Padding.

Fixed an issue with inconsistency in paged search results

Fixed | DS-46808 | PingDirectory, PingDirectoryProxy

Fixed an issue where PingDirectoryProxy could have returned an inconsistent number of entries for paged search requests. Now, to ensure consistency in the returned entries, PingDirectoryProxy sends each paged search request to one server.

Fixed an encoding issue with UTF-8 in URI search filters

Fixed | DS-48300 | PingDirectory, PingDataSync

Fixed an issue where PingDataSync couldn't properly encode certain UTF-8 characters used in a URI search request filter sent to an external server. The server is now able to encode filter values that include any UTF-8 characters.

Fixed an issue with syncing modified PingOne attributes

Fixed | DS-48669 | PingDataSync

Fixed an issue where syncing from a PingOne sync source using an attribute synchronization mode of `modified-attributes-only` resulted in changed attributes not being properly synced over.

Fixed an issue with VLV indexes and extensible match filters

Fixed | DS-48026 | PingDirectory

Fixed an issue that could have prevented the server from using VLV indexes defined with certain kinds of extensible match filters, including those using the `jsonObjectFilterExtensibleMatch` or `relativeTimeExtensibleMatch` matching rules.

Fixed an issue with inconsistent `entryUUID` values across servers

Fixed | DS-48678, DS-48720 | PingDirectory

Fixed an issue where MODDN operations on replicated PingDirectory servers configured with Groovy-scripted or third-party type password generators or validators could result in inconsistent `entryUUID` values for the same entry on different servers.

Fixed an issue with attribute value duplication

Fixed | DS-48585 | PingDirectory

Fixed an issue where replace operations that targeted attributes with subordinate types would cause the subordinate attribute values to be duplicated.

Fixed a replication issue with an `Invalid host` error

Fixed | DS-48311 | PingDirectory

Fixed an issue where disabling replication with a missing hostname sometimes caused **`dsreplication status`** to fail with an `Invalid host` error.

Fixed a configuration change issue when replacing profiles

Fixed | DS-45783 | PingDirectory, PingDirectoryProxy, PingDataSync

Resolved an issue where running the **`manage-profile replace-profile`** command could cause **`dsconfig`** changes to be made out of order.

Fixed an issue with an encryption alarm

Fixed | DS-46533 | PingDirectory

Fixed an issue where the Strong Encryption Not Available Gauge had a value of `INDETERMINATE` and showed an alarm, even when the JVM supported strong encryption. Also changed the name of this gauge to Strong Encryption Available to avoid confusion in the event of an alarm being raised.

Fixed an issue with the PSA updating the wrong entries

Fixed | DS-48358 | PingDataSync

Fixed an issue where the PSA could update incorrect entries upon a password change if there were users with the same sAMAccountName in a forest.

Fixed an issue with entry modification in replication

Fixed | DS-48491 | PingDirectory

Fixed an issue that could prevent a modify request from adding real attribute values to a replicated entry that already had one or more virtual values for that attribute.

Fixed an issue with indexing entries while debugging

Fixed | DS-48723 | PingDirectory

Fixed an issue where an untrusted composite index would prevent entries matching that index from being added or modified if a debug log publisher was enabled for the composite index.

Fixed an error message in the Delegated Admin report

Fixed | DS-48774 | PingDirectory, PingDirectoryProxy

Removed a stack trace from the error message returned when generating a Delegated Admin report with an invalid SCIM filter.

Fixed a null pointer exception in replication

Fixed | DS-48796 | PingDirectory

Fixed an NPE error that could occur when running the `dsreplication enable` command in interactive mode.

Fixed an issue with installing PingDirectory in FIPS mode

Fixed | DS-48834 | PingDirectory

Resolved an issue where installing the PingDirectory server in FIPS-compliant mode would sometimes fail with an error stating that a configuration file entry had the same DN as another entry already read from that file.

Fixed a rare startup error related to replication and sleep values

Fixed | DS-48897 | PingDirectory

Fixed a rare issue where the server could have experienced an `IllegalArgumentException` on startup due to a negative sleep value when one or more replication servers wasn't online.

Support for the HashiCorp Vault secrets engine

Issue | DS-49305 | PingDirectory

Currently, the PingDirectory server only supports version 1 of the HashiCorp Vault KV secrets engine. Learn more about KV version 1 in the Vault KV secrets engine documentation.

PingDirectory suite of products 10.0.0.6 (June 2025)

Fixed a server installation issue with Java 17 and Red Hat

Fixed | DS-49716 | PingDirectory, PingDirectoryProxy, PingDataSync

We fixed an issue that could prevent installing or running servers using Java 17 or later on Red Hat Enterprise Linux (RHEL) systems when the operating system itself is configured to run in FIPS-compliant mode.

This operating system setting is unrelated to whether the PingDirectory server is set up to run in FIPS-compliant mode.

Fixed a `userRoot` issue for server upgrades

Fixed | DS-49281 | PingDirectory

We fixed an issue that caused some server upgrades to fail. During an upgrade, the update tool added `userRoot` entries for inverted static group support to the server configuration after the `userRoot` backend had been removed.

Fixed the failure behavior for `dsreplication initialize`

Fixed | DS-49890 | PingDirectory

We fixed an issue where a PingDirectory server would continue sending binary data to the destination server after a failed attempt to initialize using `dsreplication initialize`. This behavior interfered with further initialization attempts from any other server.

Removed the restart prompt when changing a certificate alias

Fixed DS-45174 PingDirectory, PingDirectoryProxy, PingDataSync

We removed the prompt to restart the LDAP connection handler component after changing the `ssl-cert-nickname` configuration property because a restart isn't required.

Fixed a SCIM 2.0 PUT issue with attribute values

Fixed DS-49619 PingDirectory

We fixed an issue where SCIM 2.0 PUT operations involving multivalued complex attributes would incorrectly remove some of the values.

Fixed a SCIM issue with modifying `ds-pwp-modifiable-state-json`

Fixed DS-49781 PingDirectory

We fixed an issue where SCIM requests that attempted to modify the `ds-pwp-modifiable-state-json` attribute would fail.

Changed proxy transformation requirements for mapped attributes

Fixed DS-48958 PingDirectoryProxy

We updated the attribute mapping proxy transformation to require that both the source and target attribute types are defined in the local schema.

This change ensures that the server uses the correct logic when interacting with values of those attributes (for example, to identify whether the attribute type is declared as single-valued or multivalued so that it can properly format the values in REST API responses). The server now prevents adding a new instance of this proxy transformation if either of the attribute types isn't defined in the schema. It also logs a warning message on startup if any existing instance of the transformation references an undefined attribute type.

Excluded some password attributes from sync sources

Fixed DS-49212 PingDataSync

We changed the **`resync`** tool to exclude `unicodePwd` automatically from AD sync sources and `password` from PingOne sync sources.

By design, the **`resync`** tool updates the existing values for included attributes at the destination to match what's found at the source. If **`resync`** can't retrieve an attribute value at the source, it removes any existing values at the destination. Because **`resync`** can't retrieve these password attributes from their sources, we've excluded them from the attributes for **`resync`** consideration, to avoid disrupting the values at the destination.

You can still include these attributes manually in a **`resync`** operation by providing the `--includeSourceAttr` argument.

Fixed an issue with character encoding for PingOne sync destinations

Fixed DS-49362 PingDataSync

We fixed an issue where an empty space character didn't get properly encoded when URLs were sent to PingOne sync destinations.

PingDirectory suite of products 10.0.0.4 (October 2024)

Changed replication to prevent lockdown for missing changes from obsolete replicas

Improved DS-49070 PingDirectory

Changed replication behavior to prevent server lockdown for missing changes due to obsolete replicas. This change affects the following scenarios where, previously, these types of missing changes triggered lockdowns:

- The `replication-purge-obsolete-replicas` global configuration property is set to false.
- Not all servers in the topology support configurable missing changes.
- The remote server indicates lockdown for replicas that are actually obsolete.

Made it easier to upgrade replicated servers to version 10.0.0.4 or later

Improved DS-48798, DS-49090 PingDirectory

When upgrading a pre-9.2 PingDirectory server in a replicated topology to version 10.0.0.4 or later, the **`update`** tool will automatically set `replication-purge-obsolete-replicas` to `false` for that server, if not already explicitly configured.

This change helps avoid unintended consequences when upgrading a pre-9.2 replicated server, as the `replication-purge-obsolete-replicas` configuration property has a value of `true` by default in version 9.2.

After upgrade, the update tool also displays a message with more information:

```
In the 9.2.0.0 release, the implicit default value for the 'replication-purge-obsolete-replicas'
global configuration property changed from 'false' to 'true'. However, it should generally only
be set to true if all servers in the topology are at version 9.2.0.0 or later. Because this server
is being updated from a pre-9.2.0.0 version, it is possible that there are still other pre-9.2.0.0
servers in the topology. As such, the 'replication-purge-obsolete-replicas' property will be explicitly
set to false for this server if it was not explicitly set. Once you have completed the upgrade across all
servers in the topology so that there are no more pre-9.2.0.0 replicas, consider manually setting this
property to 'true' on all servers.
```

Reduced the retrieval time for the percentage of undeletable files

Improved DS-45172 PingDirectory

Used caching to speed up the retrieval of the percentage of undeletable database files for the Database Environment monitor entry.

Fixed a `config-diff` error

Fixed | DS-49071 | PingDirectory, PingDirectoryProxy, PingDataSync

Fixed an issue where **`config-diff`** would result in an `Unknown property` error when comparing configuration objects of different types.

Fixed a server startup issue

Fixed | DS-49121 | PingDirectory, PingDirectoryProxy, PingDataSync

Fixed an issue that could prevent the server from starting when configured to use a third-party key manager provider created using the Server SDK.

Removed suppression messages for disabled alerts

Fixed | DS-49119 | PingDirectory, PingDirectoryProxy, PingDataSync

Fixed an issue where alert types that were disabled would still output suppression messages.

Changed the REST API to use alternate authorization IDs

Fixed | DS-49195 | PingDirectoryProxy

Fixed an issue where the Directory REST API didn't use alternate authorization identities in entry-balanced proxy environments.

PingDirectory suite of products 10.0.0.3 (July 2024)

Increased replication speed

Improved | DS-48826 | PingDirectory

Increased throughput for replicated operations.

Reduced the size of replication monitor messages

Improved | DS-48058 | PingDirectory

To reduce the size of replication monitor messages, the `include-all-remote-servers-state-in-monitor-message` global configuration property is now set to false by default. Servers no longer include information about other remote servers in their monitor messages, but each server describes itself with its own monitor message.

Supplied missing replication error information

Fixed | DS-48785 | PingDirectory, PingDirectoryProxy, PingDataSync

Fixed an issue where **`dsreplication enable`** didn't print error information if the tool failed to establish a connection to a source or target server.

Fixed a configuration change issue when replacing profiles

Fixed | DS-45783 | PingDirectory, PingDirectoryProxy, PingDataSync

Resolved an issue where running the `manage-profile replace-profile` command could cause `dsconfig` changes to be made out of order.

Fixed an issue with syncing modified PingOne attributes

Fixed | DS-48669 | PingDataSync

Fixed an issue where syncing from a PingOne sync source using an attribute synchronization mode of `modified-attributes-only` resulted in changed attributes not being properly synced over.

Fixed an issue with inconsistent index metadata

Fixed | DS-48969 | PingDirectory

Fixed an issue that could cause an inconsistency in the metadata for a composite index record. This inconsistency could cause:

- Validator error messages in the server's error log
- Error responses to some server requests
- Failure to bring the affected backend online (rare)

In addition, the server now has added resiliency against these kinds of issues, with a better ability to identify the correct result set and notify administrators of the issue.

Fixed a null pointer exception in replication

Fixed | DS-48796 | PingDirectory

Fixed an NPE error that could occur when running the `dsreplication enable` command in interactive mode.

Fixed an issue with inconsistent `entryUUID` values across servers

Fixed | DS-48678, DS-48720 | PingDirectory

Fixed an issue where MODDN operations on replicated PingDirectory servers configured with Groovy-scripted or third-party type password generators or validators could result in inconsistent `entryUUID` values for the same entry on different servers.

Fixed an issue with VLV indexes and extensible match filters

Fixed | DS-48026 | PingDirectory

Fixed an issue that could prevent the server from using VLV indexes defined with certain kinds of extensible match filters, including those using the `jsonObjectFilterExtensibleMatch` or `relativeTimeExtensibleMatch` matching rules.

PingDirectory suite of products 10.0.0.2 (March 2024)

Added logging history for the `setup` tool

Improved DS-47831 PingDirectory

A copy of the **`setup`** script output is now saved to an archive file in the `/history` directory. This should help with troubleshooting installations where multiple server images have been extracted on top of each other and **`setup`** has been run multiple times.

Fixed an encoding issue with UTF-8 in URI search filters

Fixed DS-48300 PingDirectory, PingDataSync

Fixed an issue where PingDataSync couldn't properly encode certain UTF-8 characters used in a URI search request filter sent to an external server. The server is now able to encode filter values that include any UTF-8 characters.

Fixed an issue with attribute duplication

Fixed DS-48585 PingDirectory

Fixed an issue where replace operations that target attributes with subordinate types would cause the subordinate attribute values to be duplicated.

PingDirectory suite of products 10.0.0.1 (January 2024)

Fixed a memory issue introduced in 10.0 that could have caused the server to crash

Fixed DS-48599 PingDirectory

We fixed an uncommon issue that was causing memory usage to spike, possibly crashing the PingDirectory server.

With this issue present, when clients performed atypical `modify` operations, they might have populated entries with duplicate attribute values. If clients repeated these modifications, over time, the duplicate attribute values could have caused the server to consume a substantial amount of memory, which might have eventually caused the server to shut down with an out-of-memory error.

PingDirectory suite of products 10.0.0.0 (December 2023)

What's new in the PingDirectory 10.0 suite of products?

New

PingDirectory

- Historically, LDAP servers favor data integrity over resiliency. However, given the growth in customer topologies, there is a strong requirement for maintaining production server uptimes to meet customer expectations. In this environment, servers can be removed from the topology frequently, and if the server is down longer than the configured replication purge delay, problems could arise once the server is brought back online. In this release, a new feature allows you to configure the level of availability when encountering this issue during topology management.

- Static groups, which are the simplest and most commonly used type of group, explicitly list the DNs of group members. Server performance when adding or removing members from a static group depends partially on the group size itself, but we have identified a number of further inefficiencies in how the server handles static group membership changes. This release includes changes to improve performance when updating static groups.

  This release also introduces a new group type: inverted static groups. As with traditional static groups, inverted static group membership is explicitly defined rather than automatically determined. However, instead of storing the entire list of members in the group entry, each user entry lists the set of inverted static groups in which that user is a member. Inverted static groups with a large number of members can be more efficient to maintain than traditional static groups, because the change needed to add or remove a user only requires updating the user entry, which isn't affected by the number of members in the group. The server also provides an optional plugin that allows an inverted static group to be updated as if it were a traditional static group, intercepting attempts to alter the membership attribute in the group entry itself and making the corresponding changes in user entries instead.

- PingDirectory allows clients to interact with the server using a REST API over HTTP as an alternative to LDAP. Recent updates to the Directory REST API, including the addition of support for controls and select extended operations, have improved feature parity between the REST-based and LDAP-based interfaces, creating a more robust experience for developers using the REST API.

  While it is possible to authorize individual requests using either HTTP basic authentication (using the DN and password of the target user) or with an OAuth 2 access token obtained through another service, the Directory REST API didn't provide a fine-grained way of verifying user credentials. This release introduces a new `authenticate` endpoint, which provides a way for Directory REST API clients to verify user credentials. This enables you to better differentiate authentication failures from authorization failures, and to obtain an access token to use in authorizing subsequent requests as a specific user. Users can be identified with either a DN or a username, and the credentials may include a static password on its own or in conjunction with a delivered one-time password, a time-based one-time password, or a one-time password generated by a YubiKey device.

- PingDirectory has always offered support for defining deprecated password storage schemes. If a user successfully authenticates and provides the server their clear-text password, and if their password is currently encoded with an undesirable scheme, the server can automatically re-encode their password using a more desirable scheme. This release expands on this functionality by making it possible to re-encode passwords if the configuration of the underlying scheme has changed in a way that affects the scheme's stored representation.

  For example, if a user's password is encoded using the PBKDF2 scheme, the server can now automatically re-encode the password if their stored password uses a digest algorithm, iteration count, salt length, or derived key length that doesn't match the current configuration of that scheme. PingDirectory has also long supported the Pwned Passwords service, rejecting attempts to set passwords that are known to have been compromised. In the past, interaction with the Pwned Passwords service used a hard-coded timeout of 30 seconds in case the service became unreachable or unresponsive. You can now customize that timeout.

- PingDirectory uses the Berkeley DB Java Edition to store its data, and this database library offers support for caching some or all of the data in memory for faster access. PingDirectory also allows administrators to configure separate backends to hold different portions of the DIT. Previously, the server maintained a separate database cache for each backend, requiring the administrator to adjust the percentage of the JVM's memory that each backend is allowed to consume. This release now enables you to share a common database cache across all backends. Although this capability is disabled by default, it can simplify the server configuration by only requiring administrators to specify the total percentage of JVM memory to use for caching, without needing to configure caching separately for each backend.

- Amazon's Simple Storage Service (S3) is a popular cloud-based data storage service that can be used as a convenient off-site backup mechanism. In the past, some PingDirectory server administrators have chosen to copy certain types of files manually (for example, LDIF exports or rotated log files) to an S3 bucket as an additional layer of safety in their disaster recovery strategy. This release introduces direct support for using the S3 service as a way of backing up LDIF exports and log files.

  This release offers support for post-LDIF-export task processors. This enables you to perform additional processing automatically after successfully completing an LDIF export, including exports created as part of a recurring task. We have included an implementation that can copy the resulting export file to a specified S3 bucket for safekeeping, and it can automatically remove older export files from that bucket based on the number or age of files in that bucket. It is also possible to use the Server SDK to develop custom post-LDIF-export task processor implementations to perform other kinds of processing after an export completes.

  This release offers a new log file rotation listener that can automatically copy log files to a specified S3 bucket as soon as they have been rotated out of place. This support is available for most types of log files that the server can generate, and it also supports automatic retention based on the number or age of the files in the bucket. The server now includes a new `amazon-s3-client` command-line tool that can be used to interact with the S3 service manually. This tool can be used to manage buckets and files contained in the S3 service manually, including uploading files to or downloading files from a specified bucket.

- This release includes changes to improve performance dramatically when creating a backup, restoring a backup, or performing online replica initialization.

Fixed a security issue

Security DS-47632 PingDirectory, Delegated Admin

Fixed a security issue that could potentially affect customers using Delegated Admin. Customers are advised to apply a maintenance patch or upgrade to the latest supported version of the PingDirectory server. The Delegated Admin application is unaffected and doesn't require updating. Additional details are provided in SECADV039 (requires sign-on).

Added an `amazon-s3-client` command-line tool

New DS-47965 PingDirectory

Added a new `amazon-s3-client` command-line tool that can be used to interact with the Amazon AWS Simple Storage Service (S3) service. This tool enables you to list, create, and delete buckets, as well as list, upload, download, and delete files in a specified bucket. This may be useful in deployments where the server is configured to automatically copy rotated log files or exported LDIF files to the S3 service.

Added a request control to Directory REST API

New DS-47899 PingDirectory

Added support for access log field request control in Directory REST API requests.

Added a new `/authenticate` endpoint to the Directory REST API

New DS-47596 PingDirectory, PingDirectoryProxy

Added an /authenticate endpoint to the Directory REST API that enables users to generate an access token by providing combinations of valid credentials, depending on the authentication type specified in the HTTP request body. The supported authentication types are:

- `password`
- `passwordPlusTOTP`
- `passwordPlusDeliveredOTP`
- `passwordPlusYubiKeyOTP`

Learn more about the `/authenticate` endpoint in Directory REST API configuration.

Added five new Directory REST API endpoints to support the `/authenticate` endpoint

New DS-47641, DS-47642, DS-47644, DS-47645, DS-47646, DS-47643, DS-47648 PingDirectory

Added five new Directory REST API endpoints to support the new `/authenticate` endpoint. These endpoints enable users to interact with supporting services that facilitate the creation, delivery, and revocation of one-time passwords (OTP) and time-based one-time passwords (TOTP), which are required to perform authentication operations with the API. These endpoints include:

- `/directory/v1/{dn}/generateTOTPSharedSecret`
- `/directory/v1/{dn}/revokeTOTPSharedSecret`
- `/directory/v1/deliverOneTimePassword`
- `/directory/v1/{dn}/registerYubiKeyOTPDevice`
- `/directory/v1/{dn}/derigesterYubiKeyOTPDevice`

Learn more about these endpoints in Directory REST API configuration.

Added support for the 2b password storage variant

New DS-48119 PingDirectory

Updated the bcrypt password storage scheme to include support for the 2b variant in addition to the existing 2y, 2a, and 2x variants.

Added support for post-LDIF-export task processors

New DS-47420 PingDirectory

Added support for post-LDIF-export task processors to use in performing custom processing whenever an LDIF export task (including those invoked as part of a recurring task) successfully completes the export.

These processors include an Upload to S3 processor, which can be used to upload the resulting LDIF file to a specified Amazon S3 bucket. You can also use the Server SDK to create custom post-LDIF-export task processors. Learn more in Performing post-LDIF-export task processing.

Added support for inverted static groups

New DS-46026 PingDirectory

Added support for inverted static groups, which operate like traditional static groups in that membership is explicitly specified rather than dynamically determined, but where membership information is stored in user entries rather than in the group entry. For groups with a large number of members, inverted static groups may exhibit substantially better performance than traditional static groups.

Although it's not enabled by default, the server also provides a new plugin that allows clients to interact with inverted static groups like they interact with traditional static groups. The plugin intercepts attempts to add or remove member DNs in the group entry itself and causes the corresponding changes to be applied in the member entries. It also provides limited support for interacting with group members in the group entry for search and compare operations as if the member DNs existed in the group entries. Learn more in Using inverted static groups.

Added a `split-ldif` tool

New DS-48018 PingDirectory, PingDirectoryProxy

Added a `split-ldif` tool that can be used to split an LDIF file into multiple segments, with each having a subset of the entries below a specified base DN, and entries at or above that base DN will be included in all sets. This is primarily intended for splitting a large data set for use in entry balancing, and it offers several algorithms for dividing the entries between segments.

Added a new HTTP Connection configuration property

New DS-48055 PingDirectory

Added a new HTTP Connection configuration property to enable SNI hostname checks, which are now disabled by default.

Added a new configuration property for replication servers

New DS-47888 PingDirectory

Added the `include-all-remote-servers-state-in-monitor-message` configuration property to control whether replication monitor messages include information about remote servers. By default, the property is set to `true` so that information about remote servers is sent. Setting the property to `false` may be helpful in large topologies because the size of monitor messages scales with the number of servers.

Added a new log file rotation listener

New DS-47627

Added a new log file rotation listener that can be used to upload newly rotated log files to a specified Amazon S3 bucket. The listener can remove previously updated log files based on the specified number or age of files to retain.

Added the ability to share a single database cache

New DS-47756 PingDirectory

Added the ability to share a single database cache across all local DB backends. This is an alternative to the default behavior in which each local DB backend maintains its own independent database cache, and it can simplify cache sizing in deployments with multiple local DB backends. This behavior is controlled by two new global configuration properties:

- `use-shared-database-cache-across-all-local-db-backends` : Indicates whether to use a shared database cache. If this property is set to `true` , then all local DB backends will use a shared database cache, and you must set the property to specify the size of that shared cache. If the property is set to `false` (the default value), then each local DB backend will maintain its own independent database cache with a size specified by the `db-cache-percent property` configuration property for that backend.
- `shared-local-db-backend-database-cache-percent` : Specifies the percentage of the total JVM heap size that will be used for the shared database cache. This property will only be used if the `use-shared-database-cache-across-all-local-db-backends` property is set to `true` , in which case the server will ignore the `db-cache-percent property` in the backend configuration.

If a shared database cache is enabled, the server will expose a `Shared Local DB Backend Database Cache` monitor entry with information about that shared cache, including how much of the cache is consumed by each of the backends.

Added the `re-encode-passwords-on-scheme-config-change` property to password policy configuration

New DS-35739 PingDirectory

Added the `re-encode-passwords-on-scheme-config-change` property to the password policy configuration to indicate if the server should automatically re-encode passwords that are encoded with settings that don't match the scheme's current configuration. If a user authenticates with a mechanism that provides their password unencoded, and if the password stored in their entry is encoded with settings that don't match the current configuration for the associated password storage scheme, then the server now automatically re-encodes their password with the default password storage scheme(s) using the current settings. The following password storage schemes support this functionality: `AES256` , `ARGON2` , `ARGON2D` , `ARGON2I` , `ARGON2ID` , `BCRYPT` , `PBKDF2` , `SCRYPT` , `SSHA` , `SSHA256` , `SSHA384` , and `SSHA512` .

You can also implement this capability for custom password storage schemes developed with the Server SDK.

The `ds-pwp-state-json` virtual attribute provider has also been updated with a new `has-password-encoded-with-non-current-settings` field whose value indicates if the user's password is encoded with settings that don't match the current configuration, and a new `non-current-password-storage-scheme-settings-explanations` field that can provide additional details on how the password encoding differs from the current configuration.

Added new arguments to the `encrypt-file` tool

New DS-47612 PingDirectory

Added a `--re-encrypt` argument to the **`encrypt-file`** tool to read the contents of an existing encrypted file and re-encrypt it with a different encryption settings definition or user-supplied passphrase. If the file is currently encrypted with a user-supplied passphrase, then the `--prompt-for-current-passphrase` or `--current-passphrase-file` argument should be used to supply the current encryption passphrase. If the file is currently encrypted with an encryption settings definition, then that definition will automatically be obtained from the encryption settings database.

Added a `--find-encrypted-files` argument to the **`encrypt-file`** tool to identify encrypted files in a specified location on the filesystem. By default, the tool will search for files that are encrypted with any encryption settings definition or a user-supplied passphrase, but it can be used in conjunction with the `--encryption-settings-id` argument to only identify files that are encrypted with the specified definition.

These new arguments can be useful when migrating away from a former encryption settings definition, particularly if the former definition will eventually be removed from the encryption settings database. If a definition is removed from the encryption settings database, any files encrypted with that definition will no longer be accessible.

Added the `replication-missing-changes-policy` configuration property

New DS-45452, DS-47383 PingDirectory

Added a `replication-missing-changes-policy` configuration property for both replication servers and replication domains to control how replication handles missing changes. This property can be used to avoid missing changes lockdown in cases where such lockdown isn't beneficial to the server.

When the missing changes policy is modified, connections are restarted so that the missing changes state can be reevaluated. Lockdown mode isn't cleared, but may be cleared by running the **`leave-lockdown-mode`** tool.

Added support for an access log field request control

New DS-47557 PingDirectory, PingDirectoryProxy

Added support for an access log field request control to specify field names and values that should be included in the access log message for the associated operation.

Added support for a `generate access token` request control

New DS-47570 PingDirectory, PingDirectoryProxy

Added support for a `generate access token` request control that can be included in a bind request to indicate that the server should generate and return an access token in the bind response. That access token may be used in conjunction with the OAUTHBEARER SASL mechanism to authorize subsequent connections by that client. This can be useful in cases where the initial authentication should be performed in a manner that involves single-use credentials like a time-based one time password, a delivered one-time password, or a one-time password generated by a YubiKey device, but the client wishes to establish multiple connections in which the initial credentials can't be replayed.

Upgraded Jetty

Info DS-48071 PingDirectory

Upgraded Jetty version to 10.0.17.

Removed support for Java 8

Info DS-47558 PingDirectory

We've removed support for Java 8 in the PingDirectory server. Learn more in System requirements. Learn more about upgrading from a PingDirectory instance installed with Java 8 in PingDirectory, PingDirectoryProxy, and PingDataSync.

Removed support for two `dsreplication` subcommands

Info DS-47916 PingDirectory

Removed support for the deprecated **`remove-defunct-server`** and **`cleanup-local-server`** **`dsreplication`** subcommands. To remove a defunct server from the topology, use the **`remove-defunct-server`** command-line tool. To clean up topology references on a server, run **`remove-defunct-server --performLocalCleanup`**.

Removed the PingDataMetrics Server

Info DS-46012 PingDataMetrics

PingDataMetrics was previously deprecated and has been removed from this release. Learn more about support for versions of PingDirectory containing PingDataMetrics in Ping Identity's End-of-Life Policy (sign on required).

To monitor and provide statistics for your PingDirectory suite of products, see Monitoring PingDirectory metrics with Splunk and Monitoring server metrics with Prometheus.

Improved communication with external HTTP services

Improved DS-47454 PingDirectory

Updated the server to allow configuration of connect and response timeouts when communicating with external HTTP services, such as CyberArk Conjur and HashiCorp Vault instances, the Pwned Passwords service, and YubiKey OTP validation servers.

Updated zip compression process

Improved DS-45148 PingDirectory

To improve server performance and prevent invalid block type errors, `java.util.zip` will now be used instead of `com.jcraft.jzlib` for zip compression.

Improved how the replication generation ID is calculated

Improved DS-47695 PingDirectory

The replication generation ID, a value used by replication to determine if replicas are compatible and can be replicated, will now be calculated in a way that is independent of the disk order in which the entries are stored. This is helpful when entries are imported into new servers instead of being initialized.

Improved password security when using the Directory REST API

Improved DS-48092 PingDirectory

To increase password security when using the Directory REST API, we improved the sanitization of password-related data in API responses.

Improved server upgrade times

Improved DS-47799 PingDirectory

Improved server upgrade times by streamlining the post-upgrade stability checks.

Improved memory handling for `export-ldif` and `backup` tools

Improved | DS-44417 | PingDirectory

To help avoid excessive memory pressure on a server running multiple processes, we reduced the JVM memory requirements for the **`export-ldif`** and **`backup`** command-line tools.

Updated the `backup` tool to include a compression warning

Improved | DS-48121 | PingDirectory

To help you manage your backup and restore times, the **`backup`** tool now displays a warning when you run it with the `--compress` flag on an encrypted backend.

Updated `dsreplication` tool to avoid overwrites

Improved | DS-47820 | PingDirectory

`dsreplication` commands that produce an error are now archived to avoid being overwritten. In addition, the **`dsreplication`** command now logs subcommands in separate files.

Improved performance for backup, restore, and online replica initialization

Improved | DS-45157 | PingDirectory

Significantly improved the performance times of backup, restore, and online replica initialization processes.

Improved performance of static group updates

Improved | DS-47402, DS-47410, DS-47412, DS-47413 | PingDirectory

Improved performance when making updates to static groups.

Updated the handling of extraneous data when syncing with Active Directory

Improved | DS-46635 | PingDataSync

For Active Directory Sync sources, when setting the startpoint to `end-of-changelog`, extraneous data is no longer sent from the Active Directory server to the Sync server. With this update, setting the startpoint should be faster, particularly for slow networks.

Fixed an issue when initializing subhandlers on startup

Fixed | DS-48046 | PingDirectory

Fixed an issue where an AggregatePTAhandler's subhandlers sometimes didn't properly initialize on startup and threw a `NullPointerException`.

Fixed a logging issue when using proxied authorization

Fixed DS-48157 PingDirectory

Fixed an issue where the server didn't properly log the alternative authorization DN for multi-update extended operations that used proxied authorization.

Fixed a duplication issue when running `dsjavaproperties --initialize`

Fixed DS-45206

Fixed an issue where running `dsjavaproperties --initialize` would append duplicate arguments to `common.java-args` in the `java.properties` file.

Fixed an issue with error logging

Fixed DS-48084 PingDirectory

Fixed an issue where a `cn=config does not exist` error message would appear in the error logs after navigating to the status page of the admin console.

Fixed an issue with running `manage-profile generate-profile` on an upgraded instance

Fixed DS-47381

Fixed an issue where running `manage-profile generate-profile` on an instance that had been upgraded from an earlier version would result in a profile that contained commands that were part of the upgrade, and couldn't be used to set up new installations.

Fixed an issue with password validation

Fixed DS-47875 PingDirectory

Fixed an issue where the Dictionary password validator would sometimes incorrectly handle dictionary words contained as password substrings.

Fixed an issue that prevented use of the Changelog Password Encryption plugin in replicated environments

Fixed DS-48205 PingDirectory

Fixed an issue where the Changelog Password Encryption plugin wouldn't work properly in a replicated environment if a password was changed with a Password Modify extended operation.

Fixed issues with rootDSE search

Fixed DS-47821 PingDirectory

Fixed an issue where an ldapsearch for rootDSE didn't exclude the baseDNs that were specified in a client connection policy.

Fixed an incorrect help text suggestion when running `dsreplication initialize`

Fixed | DS-47878 | PingDirectory

Fixed an issue where help text incorrectly suggested using the `--force` flag if unable to connect to the server properly when running `dsreplication initialize`.

Fixed issues with password history

Fixed | DS-47798, DS-47898, DS-47924 | PingDirectory

Fixed an issue that could prevent the server from properly updating a user's password history for a password change if the request included the password update behavior request control, indicating that password history violations should be ignored. This control is designed to prevent the server from rejecting an attempt to change a user's password if the new password is already in the history, but it incorrectly caused the server to skip all password history processing for the update.

Fixed an issue that could cause the server to add two copies of the current password into the password history when setting a new password with the password modify extended operation. This didn't affect password changes with a regular LDAP modify operation.

Fixed an issue where the server could incorrectly allow a user to set an empty password in cases where none of the configured password validators would have rejected an empty password.

Fixed the server's handling of compact values for the `ds-cfg-allow-pre-encoded-passwords` attribute

Fixed | DS-43034, DS-47832 | PingDirectory

Fixed a regression that was introduced in the 9.3.0.0 release to allow additional values for the `allow-pre-encoded-passwords` property in the password policy configuration. This issue only affects password policies stored outside of the server configuration in local DB backends, and only those policies that include the `ds-cfg-allow-pre-encoded-passwords` attribute.

This fix enables the server to recognize and properly interpret compacted values for the `ds-cfg-allow-pre-encoded-passwords` attribute when parsing a password policy definition contained in a local DB backend. When the password policy entry is retrieved, the attribute may still appear to have a corrupt value, as the value that is actually stored in the entry would still represent the compacted token rather than the logically equivalent Boolean value. Replacing the value of the `ds-cfg-allow-pre-encoded-passwords` attribute in affected entries with the appropriate value is the best way to address this issue.

Fixed an issue with replace modifications for attributes

Fixed | DS-47975 | PingDirectory

Fixed an issue that could prevent replace modifications for attribute types with subordinate types from being properly applied.

Fixed the server's handling of SCIM patch operations including empty arrays

Fixed | DS-47790 | PingDirectory

Fixed an issue where the Configuration API treated SCIM patch operations with empty arrays as invalid. Now, the API resets configuration attributes for replace operations with an empty array and ignores add operations with an empty array.

Fixed the server's handling of search operations

Fixed DS-47585 PingDirectory

Fixed an issue that could allow the server to continue processing a search operation for longer than the allowed time limit. The server wouldn't check the time limit in the course of index processing to identify potential matching entries, and in certain cases where the server had to iterate across a very large number of index keys, the allowed time limit could be exceeded in that portion of the processing.

Fixed an issue with encryption settings initialization

Fixed DS-47784 PingDirectory

Fixed an issue where encryption settings weren't initialized before initializing password policy components when running `remove-defunct-server` against servers configured with an AES256 password storage scheme.

Fixed an issue with expensive operation logging

Fixed DS-47614 PingDirectory

Fixed an issue that caused the server to incorrectly include client certificate messages in the expensive operations log.

Fixed an issue with LDAP Connection Handler objects

Fixed DS-46312

Fixed an issue where the absence of the `request-handler-per-connection` configuration property for LDAP Connection Handler objects resulted in a single request handler being unable to acknowledge incoming client requests for long-running TLS negotiations.

Fixed the `check-replication-domains` tool requirements

Fixed DS-47655

Fixed the `check-replication-domains` tool so that the `--serverRoot` argument is no longer required, and it defaults to the server's root directory.

Fixed a missing changes error when performing replication

Fixed DS-47289 PingDirectory

Fixed a possible `NullPointerException` replication error that occurred when missing changes were found for a replica, but that replica didn't exist on all servers.

Fixed an issue with account lockout

Fixed DS-47035 PingDirectory

Fixed an issue that could prevent an unsuccessful bind attempt from being properly counted toward account lockout for a user. If the user's account had been temporarily locked as a result of too many failed authentication attempts, and if the first bind attempt after that temporary lockout period had elapsed was also unsuccessful, then the act of clearing the elapsed temporary lockout prevented the new failure from being properly recorded.

Fixed the server's handling of alerts or alarms without configuration

Fixed DS-47455

Fixed a `NullPointerException` error where an alert or alarm was raised and one or more of the alert handlers wasn't configured. This most commonly happened when the server was being stopped.

Fixed the formatting of Generic JDC sync pipe destination attributes

Fixed DS-47918 PingDataSync

We fixed an issue where, when using the **`create-sync-pipe-config`** command, the correlated attributes for Generic JDBC sync pipe destinations were a single string value. The attributes are now correctly split by commas.

Fixed an issue with syncing to Active Directory

Fixed DS-48151 PingDataSync

Fixed an issue where syncing to an Active Directory sync destination could result in the destination rejecting operations if a DN map wasn't configured on the sync class, and if the operations included modifications to the `unicodePwd` attribute.

Fixed an issue with synchronizing the `enabled` attribute in a PingOne destination

Fixed DS-47905 PingDataSync

Fixed an issue with synchronizing the `enabled` attribute of a user in a PingOne destination. This issue only occurred when attempting to enable or disable a user in PingOne from the source server.

To create an attribute mapping that modifies the enablement status of a user in PingOne, use the **`dsconfig`** tool to create a constructed attribute mapping of the following form. This ensures that the `enabled` attribute always has a well-defined value, even if the source attribute isn't present on an entry in the source server.

```
dsconfig create-attribute-mapping --type constructed --map-name mapName --mapping-name enabled --set
conditional-value-pattern:'(sourceAttribute=*) : {sourceAttribute}' --set conditional-value-pattern:'(!
(sourceAttribute=*)) : true'
```

Fixed an issue with the `manage-topology add-server` command

Fixed DS-45527 PingDataSync

Fixed an issue where a `NullPointerException` would be thrown when adding a sync server to a topology of two or more existing sync servers using **`manage-topology add-server`**.

Fixed issue with reported availability of backends

Fixed DS-48040 PingDirectoryProxy

Fixed an issue where PingDirectoryProxy didn't accurately report the availability of backends added through automatic backend discovery.

Support for the HashiCorp Vault secrets engine

Issue DS-49305 PingDirectory

Currently, the PingDirectory server only supports version 1 of the HashiCorp Vault KV secrets engine. Learn more about KV version 1 in the Vault KV secrets engine documentation.

Delegated Admin 5.0 (December 2023)

Fixed an unresponsive Save button

Fixed DS-48349 Delegated Admin

Fixed an issue where the **Save** button didn't respond when editing a user. This issue affected users configured with an auxiliary LDAP object class that required the `userPassword` attribute.

Previous Releases

Learn more about enhancements and issues resolved in previous major and minor releases of PingDirectory products using the following links:

- 9.3
- 9.2
- 9.1

Introduction to the PingDirectory server

The PingDirectory server is a high-performance, extensible Lightweight Directory Access Protocol (LDAP) directory that provides seamless data management over a distributed system while meeting the constant performance demands for today's markets.

The PingDirectory server centralizes consumer and user identity management information, subscriber data management, application configurations, and user credentials into a network, enterprise, or virtualized database environment.

The PingDirectory server can:

- Simplify administration
- Reduce costs
- Secure information in systems that scale for large numbers of users

You can find the latest information on PingDirectory server releases in the Release Notes.

Server features

The PingDirectory server provides the following features and tools.

Full LDAP version 3 implementation

The PingDirectory server fully supports LDAP v3, which supports the Request For Comments (RFCs) specified in the protocol. It provides a feature-rich solution that supports the core LDAP v3 protocol in addition to server-specific controls and extended operations.

High availability

The PingDirectory server supports N-way multi-primary replication that eliminates single points of failure and ensures high availability for a networked topology. It allows you to store data across multiple machines and disk partitions for fast replication. It also supports replication in entry-balancing proxy server deployments.

Administration tools

The PingDirectory server provides a full set of command-line tools, an admin console, and a Java-based setup tool to configure, monitor, and manage any part of the server.

The server has a task-based subsystem that provides automated scheduling of basic functions, such as:

- Backups
- Restores
- Imports
- Exports
- Restarts
- Shutdowns

The set of utilities includes a troubleshooting support tool that aggregates system metrics into a `.zip` archive, 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 PingDirectory 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
- An application administrator updating identity attributes or group membership to allow application single sign-on (SSO) access.

Learn more in the Delegated Admin Application Guide.

Security mechanisms

The PingDirectory 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 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
- SASL authentication mechanisms to secure data integrity, such as:
  - PLAIN
  - ANONYMOUS
  - EXTERNAL
  - CRAM-MD5
  - Digest-MD
  - GSSAPI

The PingDirectory server supports various providers and mappers for certificate-based authentication in addition to the ability to encrypt specific entries or sensitive attributes. Learn more in the PingDirectory Security Guide.

Monitoring and notifications

The PingDirectory server supports monitoring entries using:

- JConsole
- SNMP

- The admin 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 PingDirectory server supports configurable notifications, auditing, and logging subsystems with filtered logging capabilities. Learn more in Monitoring the PingDirectory Suite of Products.

Powerful LDAP SDK

The PingDirectory server is based on a feature-rich LDAP SDK for Java. 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, see http://www.LDAP.com.

System for Cross-domain Identity Management (SCIM) extension

The PingDirectory 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.

Learn more in Managing SCIM 1.1 and 2.0 servlet extensions.

Directory REST API

The PingDirectory server provides a REST API as the native interface for client access. Instead of trying to manage directory hierarchy or requiring attribute mapping, the Directory REST API provides direct access to directory data in a way that is dynamic, discoverable, and efficient. Learn more in the Developer portal.

Server SDK

The Server SDK 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
- PingDataSync 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 PingDirectory 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 admin console (for configuration and monitoring). The PingDirectory server provides plugins to extend the functionality of its components.

Server tools location

The PingDirectory 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.

The PingDirectory server, admin console, and LDAP SDK for Java are distributed in `.zip` format. After extracting the file, you can access the **`setup`** utility in the server root directory, located at `PingDirectory`.

Before installing the PingDirectory server, refer to Preparing the operating system (Linux) for important information on setting up your machines. Refer to Installing the PingDirectory server 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. You can find information on modifying the configuration of a server instance or a group of servers using the command-line tools and the admin console in Configuring the PingDirectory server.

Introduction to the PingDirectoryProxy server

The following overview describes the PingDirectoryProxy server's features and capabilities.

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 PingDirectory server directly or going through the PingDirectoryProxy server.

The PingDirectoryProxy server provides the following features:

High availability

The PingDirectoryProxy 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 server can follow them on the client's behalf.

Data mapping and transformation

The PingDirectoryProxy server can perform distinguished name (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 PingDirectory server, either because of changes that have occurred at the data layer or to inherent design limitations in the clients.

Learn more in Configuring proxy transformations.

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 PingDirectoryProxy server can automatically split the data across multiple systems. This feature allows the PingDirectoryProxy server to improve scalability and performance of the PingDirectory server environment.

Load balancing and failover

You can spread the workload across multiple directory servers 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 directs to a different server.

Learn more in Configuring load balancing.

Security and access control

The PingDirectoryProxy server can add additional firewall capabilities as well as constraints and filtering to help protect the PingDirectory server from attacks.

You can use a PingDirectoryProxy server in a DMZ as opposed to allowing clients to directly access the PingDirectoryProxy 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 through a VPN no matter what their identity or permissions.

Tracking of operations across the environment

In the past, administrators have 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 PingDirectoryProxy server contains controls that allow administrators to track requests back to the client that issued them.

Whenever the PingDirectoryProxy server forwards a request to the PingDirectory server, it includes a control in the request so that the PingDirectory server's access log has the IP address of the client, address and connection ID of the PingDirectory server. In the response back to the client, it similarly includes information about the PingDirectory server that processed the request, such as the connection ID and operation ID. This feature makes it easier for administrators to monitor in their environment.

Monitoring and management tools

Because the PingDirectoryProxy server uses many of the components of PingDirectory, it can leverage them to provide protocol support, logging, management tools for configuration and monitoring, and schema. You can use the **`dsconfig`** tool and the admin console to manage the PingDirectoryProxy server.

Learn more in Managing the PingDirectoryProxy server.

Introduction to the PingDataSync server

The PingDataSync server uses a dataless approach that synchronizes changes directly from the data sources in the background so that applications can continue to update their data sources directly. PingDataSync doesn't store any data from the endpoints themselves, which reduces hardware and administration costs. The server's high-availability mechanisms allow for easy failover from the main PingDataSync server to redundant instances.

PingDataSync server features

PingDataSync is designed to run with little administrative maintenance and includes the following features:

- High performance and availability with built-in redundancy.
- Dataless virtual architecture for a small-memory footprint and ease of maintenance.
- Comprehensive and straightforward setup that enables mapping attribute names, values, and distinguished names (DNs) between endpoints. For directory server endpoints, this enables schema and directory information tree (DIT) changes without custom coding and scripting.
- Multi-vendor directory server support, including:
  - PingDirectory server
  - PingDirectoryProxy server
  - Sun Directory Server Enterprise Edition
  - Sun Directory Server
  - Oracle Unified Directory
  - OpenDJ
  - Microsoft AD
  - Generic LDAP directories
- RDBMS support, including Oracle Database and Microsoft SQL server systems.
- Proxy server support including PingDirectoryProxy server.
- A notification mode that sends real-time change notifications to the application or service of your choice through the server SDK.

Installing the PingDirectory Suite of Products

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

This guide describes how to install the PingDirectory server and its add-ons, PingDirectoryProxy, Delegated Admin, and PingDataSync.

It contains instructions for preparing your environment for installation, obtaining or upgrading a Ping Identity license key, downloading the installation packages, and installing the servers. It also describes how to uninstall and upgrade the servers, and how to start, stop, and restart the servers.

System requirements

The PingDirectory server requires certain software packages to operate properly.

For optimal performance, the server requires Java for 64-bit architectures. To view the minimum required Java version, 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.

The following system requirements 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.

CPU and memory

When sizing an instance of the PingDirectory server, provide at least the following minimum specs:

- 2 processor cores
- 4 GB of RAM

You can find more information about memory allocation in Memory allocation and database cache or Finding the optimal heap size (requires authentication).

Running PingDirectory on Amazon Web Services (AWS)

To take advantage of snapshots for backing up any important files, install the PingDirectory server on an Elastic Block Store (EBS) file system. You should also move any databases, logs, and other files to a high-performance file system that's separate from the rest of the server image and create an S3 bucket for storing backups, schema, and other artifacts.

The following guidelines and considerations apply to AWS environments:

- PingDirectory doesn't support the Elastic File System (EFS) for database persistence. For the best performance, use the local SSD.
- PingDirectory has been qualified to run on the Graviton 2 processor architecture.
- Using network load balancers (NLBs) can lead to severed connections, which could cause issues that are difficult to diagnose.

Platforms

The PingDirectory, PingDirectoryProxy, and PingDataSync servers support actively maintained versions of the following operating systems:

- Amazon Linux
- Canonical Ubuntu LTS
- Oracle Linux
- Red Hat Enterprise Linux ES
- Rocky Linux
- SUSE Enterprise Linux
- Microsoft Windows Server 2022

Docker

To deploy the PingDirectory server using Docker, you must have a supported version of Docker. The following versions of Docker are supported:

- Docker 20.10.9

Learn more in the pingidentity/pingdirectory Docker image and the Ping Identity DevOps documentation.

> **Note**
>
> Only the PingDirectory 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 or 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 Java Support Policy applies to your Java Runtime Environment (JRE). You must have one of the following versions of the Java Development Kit (JDK) installed before installing the server:

- Oracle JDK 17 and 21
- OpenJDK 17 and 21

> **Important**
>
> Due to the number of supported JDK variants, we don't explicitly list all of them here. The criteria for supported JDKs are:
>
> - They must be built from the OpenJDK codebase.
>   - Supported OpenJDK distributions include, but aren't limited to:
>   - AdoptOpenJDK/Eclipse Temurin Java Development Kit (Adoptium)
>   - Amazon Corretto
>   - Azul Zulu
>   - Red Hat OpenJDK
> - They must use the HotSpot JIT compiler. They can't use OpenJ9 or other JIT implementations.

Browsers

The following browsers are supported by the PingDirectory server admin console:

- Chrome
- Firefox
- Internet Explorer 11 (and later)

Installing Java

The PingDirectory server requires Java. For optimized performance, use Java for 64-bit architectures. See System requirements for the list of supported Java versions.

> **Important**
>
> Support for Java 11 has been deprecated and will be removed in a future release.

About this task

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 server.

> **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.

To install Java, go to the website for the type of Java you want to install, locate the preferred version, and follow the instructions.

Preparing the operating system (Linux)

Several operating system customizations can improve server performance on Linux systems.

These customizations include:

- Configuring the file descriptor limits

- Tuning the file system
- Setting the maximum user processes
- Editing OS-level environment variables
- Installing sysstat and pstack (Red Hat)
- Installing dstat (SUSE Linux)
- Disabling file system swapping
- Adjusting system memory allocation
- Omitting the `vm.overcommit_memory` property
- Managing system entropy
- Setting file system event monitoring (**`inotify`**)
- Tuning the I/O scheduler

Configuring the file descriptor limits

The operating system default file descriptor limits restrict the number of PingDirectory server connections. You can change the descriptor limits to allow more connections.

About this task

The PingDirectory server allows for an unlimited number of connections by default, but the file descriptor limit on the operating system restricts 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 that you can make if you want to make sure the server shuts down safely before 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 that 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
```

Result

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.

Navigate to the type of tuning you want to perform for the relevant steps.

Setting the file system flushes

Reduce the span between file system flushes to improve PingDirectory server performance.

About this task

By default, Linux servers running the ext3 file system flush the data to disk every five seconds.

Steps

1. If the PingDirectory 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 in 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.

  Choose from:

  - 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 PingDirectory 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.

  Example:

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

- Set the *NUM_USER_PROCESSES* environment variable to `16383`.
- In the `config/num-user-processes` file, set *NUM_USER_PROCESSES* to `16383`.

Editing OS-level environment variables

Certain environment variables might affect the PingDirectory 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 PingDirectory 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 the **`sysstat`** and **`pstack`** packages. These packages 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)

The system monitoring tool **`dstat`** can help you troubleshoot PingDirectory servers on SUSE distributions.

About this task

The **`collect-support-data`** tool uses the **`dstat`** utility for system monitoring. You can obtain **`dstat`** from the OpenSUSE project website. The following process shows how to install the **`dstat`** utility.

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`**.

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:
    1. Clone the existing performance profile.
    2. Run **`tuned`**.
    3. Add the line `vm.swappiness = 0` to the file `/usr/lib/tuned/profile-name/tuned.conf`.
    4. To select the updated profile, run **`tuned-adm profile customized_profile`**.
    5. Restart the system to apply the changes.

Adjusting system memory allocation

You can adjust system memory allocation to improve overall performance by setting the `/proc/sys/vm/max_map_count` kernel tuning parameter. Doing this can prevent problems such as the slowing down of batch execution and a continuous increase of memory used by the JVM.

About this task

A good setting to use is four times the number of megabytes of system memory. For example, if you're running on a system with 128 gigabytes of memory, then calculate (128*1024=131072 megabytes) times 4, which is 524288.

Steps

1. Sign on as root user.
2. Add the line `vm.max_map_count = <megabytes>` to the file `/etc/sysctl.conf`. For example:

```
vm.max_map_count = 524288
```

3. Restart the system to apply the change.
4. (Optional) If you need to tune performance further after setting the `max_map_count` parameter, do the following:
    1. Clone the existing performance profile.

2. Run the `tuned` command.
3. Add the line `vm.max_map_count = <megabytes>` to the file `/usr/lib/tuned/profile-name/tuned.conf`.
4. To select the updated profile, run `tuned-adm profile customized_profile`.
5. Restart the system to apply the changes.

vm.overcommit_memory property">

Omitting the `vm.overcommit_memory` property

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

- For best results, monitor the `kernel.random.entropy_avail` in `sysctl` value.

inotify)">

Setting file system event monitoring (`inotify`)

You can configure an event monitoring tool such as `inotify` 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

Important

- For file systems running on an SSD or in a virtualized environment, use the recommended **`noop`** scheduler.
- For additional considerations related to network attached storage, see Databases on storage area networks, network-attached storage, or running in virtualized environments.
- 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. Restart the system to apply the change.

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 non-privileged 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.

   Example:

   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
   ```

Obtaining a Ping Identity license key

License keys are required to install, update, and renew all Ping Identity products.

How to obtain a license

To obtain a license key, contact your account representative or use the Ping Identity licensing portal.

When 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 8.3 to 9.0. When cloning a server instance with a valid license, you do not need a new license.

> **Note**
>
> The upgrade 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 admin 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:

- Use the server's `status` tool.
- In the admin console, go to the **Status** page and search for **License** on the **Monitors** tab.

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 admin console. Learn more in Upgrading a PingDirectory license.

> **Note**
>
> An expiring license causes alerts and alarms but does not affect the functionality of the product.

Upgrading a PingDirectory license

You can upgrade the license used by the PingDirectory server and any of its plugins with the **`dsconfig`** command-line tool and the admin console.

This topic assumes you already have a valid, new license file that you want your server to use.

Upgrading a license using the dsconfig command

Follow these instructions to upgrade your license using the **`dsconfig`** command in interactive or non-interactive mode. The **`dsconfig`** command is located in `<server-root>/bin`.

In interactive mode

Steps

1. Run the **`dsconfig`** command-line tool.
2. Select **License** from the **Topology** section of the menu.
3. Select **View and edit the license** from the menu.
4. Enter `1` to edit the `directory-platform-license-key` property.

   You should see a message like the following:

   ```
   The value of 'directory-platform-license-key' has been written to the file below.  You can edit and
   save the file,
   return here, and choose 'finish' to import the changes:

   /Users/<your_id>/PingDirectory/tmp/dsconfig-5614232350526437974-license-directory-platform-license-
   key.lic
   ```

5. Open the new license file in a text editor.
6. Overwrite the contents of the temporary file specified by the **`dsconfig`** tool with the contents of the new license.
7. Save your changes.

In non-interactive mode

Steps

1. Enter the following command, providing the path to the new license and making sure to preserve the double quotes:

```
dsconfig set-license-prop \
  --set "directory-platform-license-key:<path_to_new_file.lic>"
```

2. Confirm that the command completed successfully.

Upgrading a license using the admin console

Steps

1. Sign on to the admin console.
2. In the **Topology** area, click **License**.
3. Click **Load From File...**.
4. Select the new license file, and click **Open**.
5. Save your changes.

Downloading the installation packages

Download the latest PingDirectory installation packages from the downloads page on the Ping Identity website.

About this task

To download PingDirectory and its add-ons, you must have an active license and be signed on to the Ping Identity website with the email address used to obtain the license. Add-ons include PingDirectoryProxy, PingDataSync, the PingData Server SDK, and Delegated Admin.

Steps

1. Sign in to the Ping Identity website.
2. Go to the relevant PingDirectory page.

   Choose from:

   - For the latest version, go to the downloads page.
   - For previous versions, go to the previous releases page.
3. Click **PingDirectory** to download the server's `.zip` distribution file.
4. If you want to install one or more add-ons:

   Choose from:

   - For the latest versions, click the **Add-ons** tab on the downloads page.
   - For previous versions, locate the add-ons grouped with the version of PingDirectory on the previous releases page.
5. Click the add-ons that you want to download.

6. Extract the compressed PingDirectory `.zip` file to a directory of your choice. For example:

```
$ unzip  {pingdir}-<version>.zip
```

7. Extract the compressed `.zip` files for any add-ons that you downloaded.

8. For Delegated Admin, copy the folder named `/delegator` and its contents to the appropriate directory as shown in 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

Pre-installation considerations

After you have unzipped the PingDirectory server distribution file, you might 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 can 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 restart 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 before 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

You can enable encryption for directory data, backups, LDIF exports, and log files 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 data center 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 accept invalid access control instructions (ACIs) because of a less restrictive application of 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 might not offer the intended access protection. Rather than unexpectedly exposing sensitive data, PingDirectory rejects any ACIs that it cannot interpret, which ensures data access is properly limited as intended but can cause problems when migrating data with existing access control rules to a PingDirectory server.

If you are migrating from a Sun/Oracle deployment to a PingDirectory server, the PingDirectory server provides a `validate-acis` tool in the `bin` directory (UNIX or Linux systems) or `bat` directory (Windows systems) that identifies any ACI syntax problems before migrating data. Learn more in 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 doesn't write or require any data outside of the server-root directory.
> After you run the `setup` tool, do not copy the server-root to another location or system to duplicate the installation, because this isn't a supported method of deployment. If a host or file system change is needed, you can move the server-root to another host or disk location.

Installing the servers

The `setup` tool allows you to quickly install and configure a stand-alone server instance.

Server installation modes

You can run the tool in either of the following modes:

Interactive command-line mode

This mode prompts for information during the installation process. To start the installation in this mode, run the `setup` command without any options.

Non-interactive command-line mode

This mode is designed for setup scripts to automate installations or for command-line usage. To start the installation in this mode, run the `setup` command with the `--no-prompt` option as well as the other arguments required to define the appropriate initial configuration.

> **Note**
>
> Make sure you perform all installation steps while logged on to the system as the user or role under which the server will run.

> **Important**
>
> Support for Java 11 has been deprecated and will be removed in a future release.

Installation instructions

See the instructions in the following topics to install the PingDirectory suite of products:

- Installing the PingDirectory server
- Installing the PingDirectoryProxy server
- Installing Delegated Admin
- Installing the PingDataSync server

Installing the PingDirectory server

After you prepare your hardware and software systems, you can set up the PingDirectory server.

You can perform the following types of installations:

- An interactive installation guides you through the setup.
- A non-interactive installation doesn't guide you through setup and requires knowledge of the appropriate CLI commands.
- A lightweight installation is appropriate for testing or demonstration purposes.
- A Windows installation includes steps for installing on the Windows operating system.

Navigate to the appropriate section to find instructions for the type of installation you want to perform.

Installing the PingDirectory server in interactive mode

The `setup` command provides an interactive text-based command-line interface to set up a PingDirectory server instance.

Before you begin

Review Pre-installation considerations.

Steps

1. Extract the distribution `.zip` archive, then go to the server root directory.
2. Run the `setup` command.

   Example:

```
$ ./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.

Example:

```
$ env JAVA_HOME=/ds/java ./setup
```

3. Read the Ping Identity End-User License Agreement, and type `yes` to continue.
4. Enter the fully qualified host name or IP address of the local host, or press Enter to accept the default.
5. Enter the distinguished name (DN) for the initial root user, or press Enter to accept the default (`cn=Directory Manager`).
6. Enter and confirm the root user password.
7. Press Enter to enable the Ping Identity services (Configuration, Consent, Delegated Admin, Documentation, and Directory REST API) and the admin console over HTTPS.

   After setup, you can enable or disable individual services and applications by configuring the HTTPS Connection Handler.
8. Enter the port on which the PingDirectory server will accept connections from HTTPS clients, or press Enter to accept the default.
9. Select the unencrypted LDAP connection setting option for this server or press Enter to accept the default (option 3).

   Choose from:

   - `Do not accept unencrypted LDAP connections`

     > **Note**
     >
     > If you select this option, skip to step 12.

   - `Accept unencrypted LDAP connections, but require StartTLS to secure all communication on those connections`
   - `Accept unencrypted LDAP connections, but optionally allow StartTLS to secure communication on those connections`
   - `Accept unencrypted LDAP connections and do not enable support for StartTLS`
10. Enter the port on which the PingDirectory server will accept connections from LDAP clients, or press Enter to accept the default.
11. Select the desired setting for enabling LDAPS, or press Enter to accept the default.

> **Note**
>
> If you do not enable LDAPS, skip to step 12.

12. Enter the port on which the PingDirectory server will accept connections from LDAPS clients, or press Enter to accept the default.
13. Select the certificate option for this server:

    Choose from:

    - `Generate self-signed certificate (recommended for testing purposes only)`
    - `Use an existing certificate located on a Java Keystore (JKS). Enter the keystore path and keystore PIN to use an existing certificate using a Java Keystore`
    - `Use an existing certificate located on a PKCS12 keystore. Enter the keystore path and the keystore PIN to use an existing certificate using use a PKCS#12 keystore`
    - `Use an existing certificate on a PKCS11 token. Enter only the keystore PIN to use the PKCS#11 token`
14. Choose the desired encryption for the directory data, backups, and log files from the choices provided:

    Choose from:

    - `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`
15. Type the base DN for the data, or accept the default base DN of `dc=example,dc=com`.
16. 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 PingDirectory server. Refer to Importing data if you want to use other options to initialize the server.
17. Choose the option to tune the amount of memory that will be consumed by the PingDirectory server and its tools.
18. Press Enter to prime or preload the database cache at startup before accepting client connections.

    Priming the cache can increase the startup time for the PingDirectory server but provides optimum performance after startup completes. 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 PingDirectory server instance is starting.
19. Enter a location name for this server.
20. Enter a unique instance name for this server.

You cannot change the name after you set it.

21. Press Enter to accept the default (yes) to start the PingDirectory server after the configuration has completed.

    Enter `no` if you want to configure additional settings or import data. Doing this keeps the server in shutdown mode.

22. Select the desired option for populating the `config/tools.properties` file during setup, or press Enter to select the default (option 1).

    Choose from:

    - `Do not populate the tools.properties file`
    - `Populate the tools.properties file with properties needed to connect to the server`
    - `Populate the tools.properties file with properties needed to connect to the server, and also include the initial root user DN as the default bind DN`
    - `Populate the tools.properties file with properties needed to connect to the server, and also include the DN and password for the initial root user DN as the default bind DN and password`

      > **Caution**
      >
      > This fourth option, which is not recommended for production environments, populates the `tools.properties` file with properties needed to connect to the server, includes the DN for the initial root user as the default bind DN, and writes the password for that user to a `tools.pin` file for use as the default bind password.

23. In the **Setup Summary** window, review your configuration details, and then select your setup option, or press Enter to select the default (option 1):

    Choose from:

    - `Set up the server with the parameters you have given`
    - `Provide the setup parameters again`
    - `Cancel the setup`

Result

If you select option 1, your PingDirectory server is configured and initialized.

Installing the PingDirectory 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. There are two ways to set up a server in non-interactive mode:

- Use the `setup` command with the required arguments.
- Use the `manage-profile setup` command to set up the server with a configured server profile. Learn more in Setting up the server with an existing encryption settings database and Server profiles.

Using the `setup` command in non-interactive mode requires that all mandatory arguments be present for each command call. If there are missing or incorrect arguments, the `setup` command fails and aborts the process. You must use a `--no-prompt` option to suppress interactive output, except for errors, when running in non-interactive mode. You must also specify the port on which the server listens for connections:

- `--ldapPort` for connections from unencrypted LDAP clients
- `--ldapsPort` for connections from TLS-encrypted LDAPs clients

Lastly, you must use the `--acceptLicense` option. To view the license, run the `bin/review-license` command.

To tune the Java Virtual Machine (JVM) to use maximum memory automatically, use the `--maxHeapSize` option. To preload the database at startup, use the `--primeDB` option.

Learn more about configuring a deployment using a truststore in Installing the PingDirectory server with a truststore.

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

Instructions for additional tasks you can perform while installing the server in non-interactive mode are provided in the following sections.

Enabling data encryption during non-interactive setup

Enabling data encryption during setup provides the strongest protection for your PingDirectory server.

About this task

Enabling encryption during setup ensures that all data written to the local DB backends, the changelog, and the replication database will be encrypted. Enabling encryption during setup also ensures that directory backups and LDIF exports are encrypted by default.

If you enable encryption after setup, then only entries created or updated after enablement will be encrypted, along with their corresponding records in the LDAP changelog and replication database. Any data and indexes that existed before enabling encryption remain unencrypted. To encrypt pre-existing local DB backends, export the data to LDIF and then re-import the LDIF file. To ensure future encryption of backups and LDIF exports, set the `encrypt-backups-by-default` and `encrypt-ldif-exports-by-default` system configuration properties to `true`.

You can enable encryption in either interactive or non-interactive setup. Learn more about enabling encryption in an interactive setup in Installing the PingDirectory server in interactive mode.

To enable encryption non-interactively:

Steps

- Run the `setup` command with one of the following arguments:

Arguments	Description
`--encryptDataWithRandomPassphrase`	Creates an encryption settings definition for you with a strong, randomly generated key. **Note** Because all instances in a topology should have the same encryption settings definitions, you should only use this argument for standalone instances or the first instance in a topology that will export its definitions to other instances.
`--encryptDataWithPassphraseFromFile`	Creates an encryption settings definition from a passphrase you specify. When using this argument, you must specify the path for the file containing the desired passphrase. If you are setting up multiple server instances, you should supply the same passphrase to ensure that definitions are consistent.
`--encryptDataWithSettingsImportedFromFile`	Imports one or more definitions from a file generated by the **`encryption-settings export`** command. When using this argument, you must specify the path for the file containing the passphrase that protects the encryption settings export.
`--encryptDataWithPreExistingEncryptionSettingsDatabase`	Uses the encryption settings definitions from an encryption settings database that was created by another server instance.

Setting up the server with an existing encryption settings database

For added convenience, you can use an existing encryption settings database when setting up the server.

About this task

Setting up the server with an existing encryption settings database offers several advantages. You can:

- Use an encryption settings database protected by an alternative cipher stream provider. Other methods for enabling data encryption during setup will create an encryption settings database that is protected by an unencrypted password stored in a local file, and anyone with access to the system during setup can decrypt that database's contents. Alternative cipher stream providers offer stronger protection.
- Enable data encryption restrictions during setup without the need to configure them later.
- Use an encryption settings database that is frozen at the time of setup without needing to freeze it later.

Note

If you provide a frozen encryption settings database with data encryption restrictions enabled, the definitions it contains are not exposed, even to server administrators.

To set up the server with an existing encryption settings database:

Steps

- Run the **`manage-profile setup`** command on a server profile with the following properties:
  - A `setup-arguments.txt` file including the `--encryptDataWithPreExistingEncryptionSettingsDatabase` argument
  - A `<server-root>/pre-setup/config/encryption-settings/encryption-settings-db` file representing the desired encryption settings database
  - The `pre-setup-dsconfig` directory including one or more **`dsconfig`** batch files containing changes needed to enable the cipher stream provider
  - Any metadata files contained in the `<server-root>/pre-setup` directory that the cipher stream provider needs to access the encryption settings database.

    The metadata files needed depend on the enabled cipher stream provider:

    - For the file-based cipher stream provider, use the file specified by the cipher stream provider's `password-file` configuration property. If `encryption-metadata-file` has a value, you must also include the file specified by that property.
    - For the Amazon Key Management Service cipher stream provider, use the file specified by the cipher stream provider's `encrypted-metadata-file` configuration property.
    - For the Amazon Secrets Manager cipher stream provider, use the file specified by the cipher stream provider's `encryption-metadata-file` configuration property.
    - For the Azure Key Vault cipher stream provider, use the file specified by the cipher stream provider's `encryption-metadata-file` configuration property.
    - For the Conjur cipher stream provider, use the file specified by the cipher stream provider's `encryption-metadata-file` configuration property.
    - For the PKCS #11 cipher stream provider, use the file specified by the cipher stream provider's `encryption-metadata-file` configuration property.
    - For the Vault cipher stream provider, use the file specified by the cipher stream provider's `vault-encrpytion-metadata-file` configuration property.

Installing the PingDirectory server with no security enabled

You can install a PingDirectory server in non-interactive mode in a production or QA environment with no security enabled.

Steps

- Extract the distribution `.zip` file and, from the server root directory, run the **`setup`** command with the `--no-prompt` option for non-interactive mode.

The following example command uses the default root user distinguished name (`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.

Example:

```
$ ./setup --no-prompt --rootUserPassword "password" \
  --baseDN "dc=example,dc=com" --acceptLicense --ldapPort 389 \
  --instancename Instance1 --location Location1
```

Installing the PingDirectory server with a truststore

You can set up the PingDirectory server in non-interactive mode 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 and, from the server root directory, run the **`setup`** command with the `--no-prompt` option 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.

  Example:

  ```
  $ ./setup --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 admin console or the **`dsconfig`** command 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 PingDirectory 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 admin console should prevent you from removing it and tell you what still references it.

Depending on your test configuration, some example configuration changes that you can make 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 memory as possible 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.By default, the Dictionary Password Validator is referenced by the Secure Password Policy and the Root Password Policy. Therefore, you must first enter the following commands to update the password policies so that they no longer reference the validator.

```
$ bin/dsconfig set-password-policy-prop --policy-name "Secure Password Policy" --remove password-
validator:Dictionary
$ bin/dsconfig set-password-policy-prop --policy-name "Root Password Policy" --remove password-
validator:Dictionary
```

Then, you can disable the Dictionary Password Validator by using the following command:

```
$ bin/dsconfig delete-password-validator --validator-name Dictionary
```

Disable the Commonly-Used Passwords Validator

The Commonly-Used Passwords Validator loads a relatively large dictionary of banned passwords into memory. By default, this validator is referenced by the Secure Password Policy and the Root Password Policy. Therefore, you must first enter the following commands to update the password policies so that they no longer reference the validator.

```
$ bin/dsconfig set-password-policy-prop --policy-name "Secure Password Policy" --remove password-
validator:Commonly-Used Passwords
$ bin/dsconfig set-password-policy-prop --policy-name "Root Password Policy" --remove password-
validator:Commonly-Used Passwords
```

Then, you can disable the Commonly-Used Passwords Validator by using the following command.

```
$ bin/dsconfig delete-password-validator --validator-name Commonly-Used Passwords
```

Tip

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

Installing the server on Windows

Use the `setup.bat` script to install the server on Windows.

Before you begin

- Make sure that you have set the *JAVA_HOME* environment variable to the location of your Java installation directory. For more information, see the "Java Runtime Environment" section in System requirements and Installing Java.
- Review Pre-installation considerations.

About this task

Complete the following steps to install the PingDirectory server.

Steps

1. Extract the distribution `.zip` file.
2. In the Windows Command Prompt or PowerShell, go to the server's root directory, and enter **setup.bat**.
3. Read the Ping Identity End-User License Agreement, and type `yes` to continue.
4. Respond to the prompts. You can enter your own values or press Enter to accept the defaults.

   If you want to see a detailed step-by-step description of the installation, see Installing the PingDirectory server in interactive mode starting with step 3.
5. In the setup summary, review your configuration details, and then select your setup option, or press Enter to select the default (option 1): Set up the server with the parameters you have given.

   Result:

   If you select option 1, your PingDirectory server is configured and initialized.
6. If you want to run the server as a Windows service, follow the instructions in Running the server as a Microsoft Windows service.

Installing the PingDirectoryProxy server

After you install the PingDirectory server, you can install and set up the PingDirectoryProxy server.

You can install using interactive mode, which guides you through the setup, or non-interactive mode, which requires that you enter the appropriate CLI commands for server setup.

Installing the server in interactive mode

The `setup` command provides an interactive text-based interface to install a PingDirectoryProxy server instance.

Installing the first server

Steps

1. Change to the server root directory.

   Example:

   ```
   cd PingDirectoryProxy
   ```

2. Run 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 PingDirectoryProxy server services (Configuration, Documentation, and Directory REST API) and admin 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 where the PingDirectoryProxy server should accept connections from HTTPS clients, or press **Enter** to accept the default.
10. Enter the port where the PingDirectoryProxy server should accept connections from LDAP clients, or 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 (JKS) or the PKCS#12 key store, the key store 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:

    Choose from:

    - `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:

    Choose from:

    - `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 PingDirectoryProxy 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 servers

About this task

The `setup` command provides an interactive text-based interface to install a PingDirectoryProxy server instance that clones a previously installed PingDirectoryProxy server instance.

Steps

1. Change to the server root directory.

Example:

```
cd PingDirectoryProxy
```

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 PingDirectoryProxy 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 PingDirectoryProxy 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 PingDirectoryProxy 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 PingDirectoryProxy 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 PingDirectoryProxy server.

```
Enter the fully qualified host name or IP address of the local host [proxy.example.com]:
```

10. Choose the location of your new PingDirectoryProxy 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 server in non-interactive mode

You can run the `setup` command in non-interactive mode to automate the installation process using a script or to run the command directly from the command line.

The following sections describe how to install the first PingDirectoryProxy server, how to install additional servers, and how to install the server with a trust store.

Installing the first server

About this task

The `setup` command 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` option is only valid if the `--entryBalancing` option is also present.

If you're using entry balancing, tune the amount of memory devoted to the PingDirectoryProxy 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.

Steps

1. Run the `setup` command with the `--no-prompt` option.

   The command uses the default root user distinguished name (DN) (`cn=Director Manager`) with the specified `--rootUserPassword` option. You must include the `--acceptLicense` , `--instanceName` , and `--location` options or the `setup` command will generate an error message.

   ```
   $ env JAVA_HOME=/ds/java ./setup --no-prompt \
   --rootUserDN "cn=Directory Manager" \
   --rootUserPassword "password" --ldapPort 389 \
   --acceptLicense \
   --instanceName ds1 --location Denver
   ```

Installing additional servers

Steps

1. Run the `setup` tool with the `--no-prompt` option.

```
$ env JAVA_HOME=/ds/java ./setup --no-prompt \
--rootUserDN "cn=Directory Manager" \
--rootUserPassword "password" --ldapPort 1389 \
--localHostName proxy2.example.com \
--peerHostName proxy1.example.com --peerPort 389 \
--peerUseNoSecurity --acceptLicense --instanceName ds1 \
--location austin1
```

Installing the server with a trust store

About this task

If you've already configured a trust store, you can run the `setup` command to enable security. The following example enables both SSL and StartTLS security. It also specifies a JKS and trust store that define the server certificate and trusted CA. The passwords for the key store files are defined in the corresponding `.pin` files, where the password is written on the first line of the file.

> **Note**
>
> The server expects the password to the private key within the key store to be the same as the password to the key store. If it isn't the same, you can define the private key password within the admin console or using the `dsconfig` command by editing the Key Manager Provider standard configuration object.

Preparing key stores, trust stores, and `.pin` files before `setup`

Running `setup` doesn't copy your key and trust store files or their associated `.pin` files to the server root. You should manually copy those files into the `config` directory of the server root, as follows:

- `config/keystore`
- `config/keystore.pin`
- `config/truststore`
- `config/truststore.pin`

> **Important**
>
> If you choose to leave these files in their original locations and supply the original reference paths when running `setup`, don't delete them afterwards. The server uses the files directly from those original locations.

Steps

1. To install a PingDirectoryProxy server with a trust store, run the `setup` command.

```
$ env JAVA_HOME=/ds/java ./setup \
  --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 \
  --instanceName ds1 --location Denver

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
```

Installing Delegated Admin

Different options and procedures are available when you install Delegated Admin depending on your environment and the location of your installation.

To install Delegated Admin, read the following sections:

- Installation requirements
- Preparing to install the application
- Installing the application
- Completing the installation

Installation requirements

Before you install Delegated Admin, make sure you meet the requirements for the installation.

Before you begin

Regardless of the location of your Delegated Admin installation, ensure 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. Learn more about upgrading a PingDirectory server in Upgrading the PingDirectory, PingDirectoryProxy, and PingDataSync servers. Learn more about installing and configuring a PingDirectory server in Installing the PingDirectory server and PingDirectory Server Administration Guide.	Refer to the Compatibility matrix.
PingFederate Server	Provides identities for authentication and authorization. Learn more about installing and configuring the PingFederate server in Configuring the PingFederate server or the PingFederate documentation.	PingFederate 9.0.0 or later

Installation locations

The location that you choose determines the steps that you must perform to install Delegated Admin.

You can install Delegated Admin in any of the following locations:

- A PingDirectory server, including replicated instances
- A PingDirectoryProxy server
- An external web server

Preparing to install the application

To install Delegated Admin on a PingDirectory server or on a replicated instance of a PingDirectory server, complete the relevant tasks in this section before installing the application.

Preparing to install Delegated Admin on a PingDirectory server

About this task

You can install Delegated Admin on a PingDirectory server with or without `Install with sample data` chosen as the installation option.

To install without `Install with sample data`, refer to step 8 in Downloading the installation packages.

To install with `Install with sample data`:

Steps

- Remove the relevant access control instruction (ACI) from the PingDirectory server base entry.

  Learn more in the `delegator/remove-sample-directory-data-aci.ldif` file.

Preparing to install Delegated Admin on a replicated instance of the PingDirectory server

About this task

To install Delegated Admin on a replicated instance of the PingDirectory server:

Steps

1. Make sure that replication is enabled for the PingDirectory servers.

   Learn more about enabling server replication in Managing replication.

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`.

   Example:

   ```
   $ 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 ACI from the PingDirectory server base entry.

   > **Note**
   >
   > Because replicated instances share the same data, perform this step against only one of the servers. Learn more in the `delegator/remove-sample-directory-data-aci.ldif` file.

4. If you're installing Delegated Admin on a PingDirectory server installation that didn't have `Install with sample data` chosen as the installation option, proceed to Downloading the installation packages.

Installing the application

The steps for installing Delegated Admin depend on whether you are setting up the application in a Unix or Linux environment or in a Windows environment.

> **Important**
>
> In all environments, if port 443 is used but not specified in the PingFederate Base URL, do not assign a value to `window.PF_PORT`.

Click the following tabs to see instructions for the type of install you want to perform.

On Unix or Linux

Unix or Linux
Steps

- Run the following script in the `/delegator` directory.

```
$ ./set-up-delegator.sh
```

Result:

The system generates:

- A configuration file, `config.js`, located in the `/webapps/delegator/app` directory.
- A batch file, `delegated-admin.dsconfig`, in the `/webapps/delegator` directory.

The `setup` tool also generates several command-line prompts for completing the Delegated Admin installation. After you provide the distinguished name (DN) of the root user for your PingDirectory instance, you're prompted to enter information regarding your PingFederate server and the hosting location of the Delegated Admin web app. Enter your own values or press Enter to accept the default values.

> **Note**
>
> If you answer `y` to the **`setup`** script question `Is Delegated Admin being installed in a topology containing PingDirectoryProxy?`, the system also generates a batch file, `delegated-admin-for-proxy.dsconfig` in the `/webapps/delegator` directory.

On Windows

Windows
Steps

1. In the Delegated Admin application directory, copy or rename the file `example.config.js` to `config.js`.

   The `config.js` file 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.

config.jsVariable	Value
`window.PF_HOST`	Public address of the PingFederate server to which the application redirects the user's browser when signing on.
`window.PF_PORT`	PingFederate port number. **Note** 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. Concatenate the following files into a single file named `delegated-admin.dsconfig`:

   - `delegated-admin-template-common.dsconfig`
   - `delegated-admin-template-ds.dsconfig`
   - `delegated-admin-template-ds-or-proxy.dsconfig`
   - `delegated-admin-template-webapp.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`.

Completing the installation

Regardless of whether you are installing Delegated Admin in a Unix or Linux or Windows environment, perform the relevant steps in this section after you complete the previous OS-specific tasks.

Navigate to the type of installation you want to complete for the relevant steps.

> **Important**
>
> After completing the installation, refer to the next steps section.

The PingDirectoryProxy server

Before you begin

> **Note**
>
> The following task assumes that when you ran the **`setup`** command, you answered `y` to the question `Is Delegated Admin being installed in a topology containing PingDirectoryProxy?`

About this task

If you are installing Delegated Admin on the PingDirectoryProxy server, you must configure the proxy instance using the **`delegated-admin-for-proxy.dsconfig`** script as described in Locations other than PingDirectoryProxy.

To configure all instances of the PingDirectory server:

Steps

1. Apply the commands from the `delegated-admin.dsconfig` batch file to all instances of the PingDirectory server.

```
$ ./bin/dsconfig \
  --bindDN "cn=Directory Manager" \
  --bindPassword <password> \
  --no-prompt \
  --batch-file webapps/delegator/delegated-admin.dsconfig \
  --applyChangeTo server-group
```

2. Apply the commands from the `delegated-admin.dsconfig` batch file to all instances of the PingDirectory server, as explained in Replicated instances of PingDirectory.

Replicated instances of the PingDirectory server

About this task

If you are installing Delegated Admin on one or more replicated instances of the PingDirectory server:

Steps

- Apply the following commands in `delegated-admin.dsconfig` to each instance:

```
$ ./bin/dsconfig \
  --bindDN "cn=Directory Manager" \
  --bindPassword <password> \
  --no-prompt \
  --batch-file webapps/delegator/delegated-admin.dsconfig \
  --applyChangeTo server-group
```

External web server

Before you begin

> **Note**
>
> The following steps assume that when you ran the **`setup`** command, you answered `n` to the question `Will the web app be hosted in PingDirectory?`

Steps

1. Open `config.js` in a text editor.
2. Change the variable values to specify the location of the PingDirectory server.

config.jsVariable	Value
`window.DS_HOST`	Host name of the PingDirectory server
`window.DS_PORT`	HTTPS port of the PingDirectory server

   To view an example outline that features these settings, see `example.config.js`.
3. Save your changes to `config.js`.
4. 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.

   Example:

   ```
   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"
   ```
5. (Optional) If you will be enabling administrators to run Delegated Admin reports in your configuration, run **`dsconfig`** with the `set-http-servlet-cross-origin-policy-prop` option.

   Example:

   ```
   dsconfig set-http-servlet-cross-origin-policy-prop \
     --policy-name "Delegated Admin Cross-Origin Policy" \
     --set cors-exposed-headers:Content-Disposition
   ```

All locations except the PingDirectoryProxy server

About this task

To continue installing Delegated Admin on a PingDirectory server or on an external web server:

Steps

- Run the following command with `delegated-admin.dsconfig` on the appropriate server:

```
$ ./bin/dsconfig \
  --bindDN "cn=Directory Manager" \
  --bindPassword <password> \
  --no-prompt \
  --batch-file webapps/delegator/delegated-admin.dsconfig
```

Next steps

> **Important**
>
> You can't sign on to Delegated Admin until you configure the rights of the delegated administrators. Learn more about configuring administrative rights, the REST resource type, session timeout values, and other properties in Configuring Delegated Admin.

After you configure Delegated Admin, verify that the application is installed and working successfully.

Verifying the installation

Steps

- To verify that Delegated Admin is installed successfully, go to `https://<webserverHost>:<httpPort>/delegator` and sign on to the application.

Troubleshooting

If your sign on attempt is unsuccessful, see Enabling log tracing and use your browser's debug feature to gain insight into the token-validation process.

Installing the PingDataSync server

This section describes how to install and run PingDataSync.

Installing the main 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.

Steps

1. Sign on as a user other than root.
2. Obtain the latest `.zip` release bundle, as described in Downloading the installation packages, 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 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, 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 admin console.
11. Enter the port on which PingDataSync will accept connections from HTTPS clients, or press Enter to accept the default.
12. Enter the port on which PingDataSync 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 set up the server with the listed parameters, change the parameters, or cancel the setup.

Next steps

After the server configuration is complete, you can run the `create-sync-pipe-config` tool configure the synchronization environment.

The admin console enables browser-based server management, the `dsconfig` tool enables command-line management, and the Configuration API enables management by third-party interfaces.

Installing a failover server

About this task

PingDataSync 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 `.zip` build. Name the unpacked directory something other than the first server instance directory.

   ```
   $ unzip PingData<server_version>.zip -d  <server2>
   ```

2. Go to the server root directory.

3. Run the `setup` command without any options to install the failover server in interactive mode, or run the following command to install it in non-interactive mode:

```
$ ./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` command. 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. To obtain the name of a particular server, run the `dsconfig` command with the `list-external-servers` option.

```
$ bin/dsconfig list-external-servers
```

6. To change the priority index of the server, run the `bin/dsconfig` command as follows:

```
$ bin/dsconfig set-external-server-prop \
  --server-name  <server2>.example.com:389 \
  --set <server>-priority-index:1
```

Uninstalling the PingDirectory and PingDirectoryProxy servers

The server provides an **`uninstall`** command-line utility for quick removal of the code base.

To uninstall a server instance, run the **`setup`** command in either 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 use this mode, run the **`bin/uninstall`** command without any options.

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 server instance, you can also manually remove the server by stopping it and recursively deleting the directory and subdirectories as shown in the following example:
>
> ```
> $ rm -rf /ds/PingDirectory
> ```

Uninstalling the server in interactive mode

Interactive mode uses the `uninstall` command, a text-based, command-line interface to help you remove your server instance.

About this task

If the `uninstall` command cannot remove all of the server files, it generates a message with a list of the files and directories that must be deleted manually. The `uninstall` command must be run as either the root user or the same user (or role) that installed the server.

Steps

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

   Example:

   ```
   $ ./uninstall
   ```

2. Select the components to be removed.

   Choose from:

   - 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 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 6.
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.

   Example:

```
How do you want to trust the server certificate for the PingDirectory 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 server is running, it 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` command provides a non-interactive method to run 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` with the `--remove-all` option to remove all of the server's libraries.

   The optional `--quiet` option suppresses output information.

   Example:

   The following command assumes that the server is standalone and not part of a replication topology:

   ```
   $ ./uninstall --remove-all --no-prompt --quiet --forceOnError
   ```

2. View the logs for any remaining files and manually remove any remaining files or directories.

Uninstalling selected components in non-interactive mode

Use the `--backup-files` with the `uninstall` command to remove the server's backup files.

Steps

1. From the server root directory, run the `uninstall` command with the `--backup-files` option.

   Example:

   ```
   $ ./uninstall --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 servers

Software builds of the servers are periodically released with new features, enhancements, and fixes for improved server performance. This section describes how to upgrade the servers in the PingDirectory suite of products.

Learn more in the following topics:

- Upgrading the PingDirectory, PingDirectoryProxy, and PingDataSync servers
- Upgrading Delegated Admin

Upgrade considerations

When upgrading your server, there are multiple upgrade scenarios with different implications that you need to consider.

PingDirectory, PingDirectoryProxy, and PingDataSync

Consider the following when upgrading the PingDirectory suite of products.

> **Important**
>
> After reviewing the following considerations, for upgrades of PingDirectory or PingDirectoryProxy, review these additional upgrade considerations.

Considerations when upgrading to 10.3.0.0 or later

Custom SDK extensions using Javax must be migrated and recompiled

Several components were upgraded in version 10.3 of the PingDirectory suite of products. If any of your custom Server SDK extensions have classes that import `javax.*` packages, you need to migrate them to the equivalent `jakarta.*` packages and then recompile the extensions.

Java considerations

Support for Java 11 has been removed, and upgrading to Version 10.3 of PingDirectory, PingDirectoryProxy, or PingDataSync will fail unless you are running Java 17 or 21.

Prerequisites for upgrading

You must meet one of the following requirements:

- Your default Java installation is a supported version.

- You are pointing one of the following environment variables to a supported version of Java:
  - `UNBOUNDID_JAVA_HOME`
  - `JAVA_HOME`

> **Note**
>
> If you use environment variables to point to your Java installation, these will override your default Java version. We recommend you set only one variable. If both are set, `UNBOUNDID_JAVA_HOME` takes precedence.

> **Important**
>
> Java 11 is no longer supported. You can't upgrade a server instance to version 10.3 without first updating Java to a supported version.

Updating from a server running a supported version of Java

If you're upgrading a server running Java 17 or 21 to version 10.3, you can proceed with the server upgrade.

Updating from a server running an unsupported version of Java

Before upgrading the server to version 10.3, you must install either Java 17 or 21. For more information, see System requirements. Upgrading to version 10.3 after updating Java requires changes to the `java.properties` file.

> **Important**
>
> You must also meet one of the prerequisites listed in the previous section before upgrading the server.

Select one of the following options for modifying the `java.properties` file. Where a Java version is specified, substitute your installed, supported Java version.

- Option 1: Before upgrading the server, convert the file manually:
  1. Edit the `config/java.properties` file to update the Java version and convert the JVM parameters to be specific to Java 17.
  2. Run the **`bin/dsjavaproperties`** command to put the changes into effect.
- Option 2: Before upgrading the server, create a new file:
  1. Rename the old `java.properties` file.
  2. Run the **`bin/dsjavaproperties`** command to initialize a new Java 17 `java.properties` file.

     For this option, run:

     ```
     bin/dsjavaproperties --initialize
     ```

  3. Upgrade the server using the generated `java.properties` file, and then restore your customized settings from the original file.
- Option 3: Allow the upgrade to replace the file:
  1. Upgrade the server to version 10.3.

The upgrade process will overwrite the `java.properties` file and the original file will be saved as `java.properties.old`. A `java.properties.change` file will also be created, containing the diff output between the new and old `java.properties` files.

2. Restore or convert the JVM settings that were overwritten during the upgrade process.

Considerations when upgrading to 9.3.0.0 or later

For service accounts that use password storage schemes with high computational processing costs (for example, PBKDF2, bcrypt, scrypt, or Argon2), the server could process bind requests much slower than in previous versions.

> **Important**
>
> The default root password policy for the PingDirectory suite of products uses the PBKDF2 password storage scheme.

Storing a password encoded with the PBKDF2 scheme can make it many times more expensive for an adversary to crack, but you can get even better protection by creating a very strong password stored with SSHA256.

You should create a separate password policy for your service account. Choose a fast but cryptographically strong password storage scheme, such as SSHA256, and set a very strong password according to NIST guidelines.

Considerations when upgrading to 9.0.0.0 or later

Because of major updates to Spring dependencies, Spring configuration properties in the admin console configuration files for the PingDirectory suite of products earlier than version 9.0.0.0 are not compatible with the admin console bundled with 9.0.0.0 and later versions. Attempting to use these older configuration files will result in the console failing to start.

If you are using older admin console configuration files, you must update them. Replace the following excerpt in the old `application.yml` file:

```
spring:
  profiles.active: default
  main.show-banner: false
  thymeleaf.cache: true
  thymeleaf.prefix: classpath:/public/app/
```

with the following:

```
spring:
  profiles.active: default
  web.resources:
    # 1 year. Update the corresponding value in MvcConfig if this changes.
    cache.period: 31536000
    add-mappings: false # use our custom mappings instead of the defaults
  main:
    banner-mode: "OFF"
  thymeleaf:
    prefix: classpath:/public/app/
```

PingDirectory and PingDirectoryProxy

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 PingDirectory server `.zip` file on the server that will be updated. In addition, you must run the **`update`** command 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`** command verifies that the Java version installed meets the new server requirements. Before running the command, install the Java version that is supported by the new server.
- For precautionary measures, back up 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`** commands 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**
>
> Learn more in Planning your upgrade in the Best Practice Guides.

Upgrade considerations introduced in PingDirectory 10.1.0.0

Keep in mind the following upgrade considerations introduced in PingDirectory 10.1.0.0.

Schema backends now have calculated generation ID values, represented by the `ds-sync-generation-id` operational attribute, rather than the fixed legacy value of `16848715`. This change prevents incompatible schema backends from replicating with each other.

Important

This change also creates a known replication issue for the following scenario:
Given a topology of newly created 10.1.0.0 or later servers (that were not upgraded from an earlier version) with schema replication active, you add a pre-10.1.0.0 server to the topology. The schema then gets initialized from a 10.1.0.0 or later server to the pre-10.1.0.0 server.
In this scenario, the schema gets copied as intended, but the 10.1.0.0 or later servers and the pre-10.1.0.0 server won't replicate schema changes after initialization, due to the non-matching schema generation IDs.
In general, you should not initialize from a newer server to an older server.

Upgrade considerations introduced in PingDirectory 9.x

Keep in mind the following upgrade considerations introduced in PingDirectory 9.x versions.

Permissive Modify no longer enabled by default in Directory REST API

Important

In previous versions, Permissive Modify behavior was enabled by default for all API modify (PATCH) requests. In version 9.3, this behavior is controlled by an `always-use-permissive-modify` configuration parameter, which now has a default value of `false`. If you see a change in your application's behavior as a result of this default configuration, you can re-enable Permissive Modify by changing the value of the configuration parameter to `true` with `bin/dsconfig`.

Enabling Permissive Modify behavior affects operations like removing an attribute that does not exist, or adding a duplicate value to a single-value attribute, by returning a "success" response without actually modifying the underlying data. Having the `always-use-permissive-modify` configuration parameter set to `false` by default will return "failure" responses for such permissive operations.

Enabling replication-purge-obsolete-replicas

Warning

The following requirement applies when upgrading from a version earlier than 9.2 to version 9.2 or higher:
You must set the `replication-purge-obsolete-replicas` global configuration property to `true` on each server in the topology before beginning the upgrade process. Failure to do so could result in a server entering lockdown mode over missing changes from an obsolete replica.
Learn more about Discovering obsolete replicas or Purging obsolete replicas.

Cleaning replication history

When cleaning replication history from a PingDirectory server, you must use the new `remove-defunct-server` argument `--performLocalCleanup`. If you have existing automation around disaster recovery, the previous method of running `remove-defunct-server` without bind credentials no longer performs this replication cleanup step. For PingDirectory versions 9.0.0.0 and later, update any existing automation to use the `--performLocalCleanup` flag.

Delegated Admin

Consider the following points when upgrading your version of Delegated Admin.

Considerations

> **Note**
>
> If you're running Delegated Admin 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 the Compatibility matrix.

Upgrade considerations introduced in Delegated Admin 4.9

The default OpenID Connect (OIDC) grant type used by the `dadmin` client has been updated to Authorization Code with PKCE. The Delegated Admin application will continue to function normally with the Implicit grant type.

If you want to switch to Authorization Code with PKCE, see Changing the default OIDC grant type.

Upgrade considerations introduced in Delegated Admin 4.8

Two new permissions that affect user resource types have been added in Delegated Admin 4.8:

- `Update-profile` grants the ability to update user profiles without allowing password-related privileges.
- `Reset-password` grants the permission to reset passwords without the ability to change other user attributes.

To preserve current admin rights, no action is required after you upgrade.

For more information, see Configuring delegated administrator rights on the PingDirectory server.

Upgrade considerations introduced in Delegated Admin 4.6

To use the functionality that allows a help desk agent to trigger a password reset for a user, you must enable the Modifiable Password Policy State plugin on the PingDirectory server that serves as a resource backend.

To enable the **Initiate Password Reset** menu option on user entries, perform the following steps:

1. Run the following command to enable the plugin needed for triggering **Initiate Password Reset**:

```
dsconfig set-plugin-prop \
--plugin-name "Modifiable Password Policy State Plugin" \
--set enabled:true --set "base-dn:${searchbasedn}" \
--set "filter:(|(objectClass=person)(objectClass=ds-cfg-user))"
```

2. Run the following command to add a Delegated Admin attribute to the users rest type for `ds-pwp-modifiable-state-json`:

```
dsconfig create-delegated-admin-attribute \
--type-name users  \
--attribute-type ds-pwp-modifiable-state-json  \
--set "display-name:Modifiable Password Policy State"  \
--set display-order-index:9999
```

Note

When you install Delegated Admin 4.6 using the `delegated-admin.dsconfig` script, the Modifiable Password Policy State plugin is enabled. If you're upgrading from a previous version of Delegated Admin, you must manually enable the plugin and add the `ds-pwp-modifiable-state-json` attribute as a Delegated Admin attribute.

Upgrading the PingDirectory, PingDirectoryProxy, and PingDataSync servers

When upgrading your server, there are multiple upgrade scenarios with different implications that you need to consider. Use the server's `update` utility to upgrade the current server code version.

Upgrading the server

Use the following steps to upgrade the PingDirectory, PingDirectoryProxy, and PingDataSync servers.

Before you begin

This task shows the upgrade process for the PingDirectory server as an example.

Steps

1. Download and extract the new version of the PingDirectory 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` command provided with the new server package to update the existing PingDirectory server.

   Example:

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

Note

The update tool might prompt for confirmation of the server configuration changes if it detects customization.

Reverting an update

After the server has been updated, you can revert to the last version or one level back using the **`revert-update`** command.

About this task

The **`revert-update`** command accesses a log of file actions taken by the updater to put the file system back to its prior state, including the prior server configuration. If you have run multiple updates, you can run the **`revert-update`** command 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.

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 aren't critical and don't appear on subsequent start ups.

To revert to the most recent server version:

Steps

- Run the **`revert-update`** command in the server root directory to revert back to the most recent version of the server.

  Example:

  ```
  $ PingDirectory-old/revert-update
  ```

Upgrading or reverting a topology

For servers in a topology, you can upgrade or revert updates by following the steps in Upgrading the server or Reverting an update on each server in the topology.

Troubleshooting **`revert-update`** *failures*

When you run **`revert-update`**, if the topology doesn't have a primary server, the command fails. In this case, you must make one of the remaining updated servers in the topology the primary. This enables the chosen instance to run the **`revert-update`** command successfully.

> **Important**
>
> For a given topology, don't set more than one server at a time as primary. After completing the revert for the topology, set `force-as-master-for-mirrored-data` back to `false` for the server you set as primary.

To make a server the primary, run the following command on the server:

```
$ bin/dsconfig set-global-configuration-prop \
  --set force-as-master-for-mirrored-data:true
```

Upgrading Delegated Admin

If you are running Delegated Admin 4.3 or earlier, upgrade it to the latest version to use PingDirectory 8.2 or later.

Before you begin

For information about the compatibility between Delegated Admin and PingDirectory server versions, see the Compatibility matrix.

Steps

1. Extract the contents of the Delegated Admin upgrade `.zip` archive.
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 the PingDirectory server by running `$ bin/stop-server --restart`

.

Starting, stopping, and restarting the server

This section contains the following topics:

- Starting the server
- Starting the server at boot time
- Stopping the server
- Scheduling a server shutdown
- Restarting the server

Starting the server

You can start the server as a background or foreground process.

Starting the server as a background process

Use a terminal window to start the server as a background process.

Steps

- To start the server as a background process on UNIX or Linux systems, run the **`bin/start-server`** command without any options.

  An analogous command is in the `bat` folder on Microsoft Windows systems.

Starting the server as a foreground process

Use a terminal window to run the server as a foreground process.

Steps

- To start the server as a foreground process, open a terminal window and enter `bin/start-server` with the `--nodetach` option.

  Example:

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

Starting the server at boot time

By default, the server does not start automatically when the system is booted. Instead, you must start it manually with the **`bin/start-server`** command.

You can use the **`create-systemd-script`** utility to configure the server to start automatically when the system boots. If you prefer, you can also create the script manually.

Starting the PingDirectory server at boot time

Steps

1. Create the service unit configuration file in a temporary location, where "ds" is the user running the server.

   Example:

   ```
   $ 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`**.

   Example:

   ```
   $ systemctl daemon-reload
   ```

4. To start the server, run the **`start`** command.

   Example:

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

5. To configure the server to start automatically when the system boots, run the **`enable`** command.

   Example:

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

6. Sign off 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 from the /etc/rc0.d directory (starting with a "K" to ensure that the server is stopped).

Starting the PingDirectoryProxy server at boot time

Steps

1. Create the service unit configuration file in a temporary location, where "ds" is the user running the server.

   Example:

   ```
   $ bin/create-systemd-script \
        --outputFile /tmp/ping-directory-proxy.service \
        --userName ds
   ```

2. As a root user, copy the **`ping-directory-proxy.service`** configuration file into the `/etc/systemd/system` directory.

3. To read the new configuration file, reload **`systemd`**.

   Example:

   ```
   $ systemctl daemon-reload
   ```

4. To start the server, run the **`start`** command.

   Example:

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

5. To configure the server to start automatically when the system boots, run the **`enable`** command.

   Example:

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

6. Sign off 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).

Starting the PingDataSync server at boot time

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. Sign on as root and move the generated `Ping-Identity-Sync.sh` script into the `/etc/init.d` directory.
3. 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-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
```

Stopping the server

The server provides the command, **`bin/stop-server`**, to stop the server. You can run it manually from the command line or within a script.

About this task

If the 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 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 following command to shut down the server.

  Example:

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

Scheduling a server shutdown

You can schedule a server shutdown using the **`bin/stop-server`** command with the `--stopTime YYYYMMDDhhmmss` option.

About this task

The following step applies to PingDirectory, PingDirectoryProxy, and PingDataSync.

Steps

- Run the **`bin/stop-server`** command with the `--stopTime YYYYMMDDhhmmss` option.

  The 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, 2023 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, which writes the reason for the shutdown to the logs.

  > **Note**
  >
  > The **`bin/stop-server`** command must be run with the `--task` argument. If this argument is not provided, an error message displays.

  Example:

  ```
  $ bin/stop-server --stopTime 20230606134500Z --task --port 1389 \
    --bindDN "uid=admin,dc=example,dc=com" --bindPassword secret \
    --stopReason "Scheduled offline maintenance"
  ```

Restarting the server

You can restart the server using the **`stop-server`** command with the `--restart` option.

About this task

Running this 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` option.

  Example:

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

PingDirectory Server Administration Guide

The PingDirectory server is a high-performance, extensible Lightweight Directory Access Protocol (LDAP) directory that provides seamless data management over a distributed system while meeting the constant performance demands for today's markets.

PingDirectory product documentation

Support

https://support.pingidentity.com

Getting started with the PingDirectory server

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

> **Note**
>
> The admin console's 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 PingDirectory server configuration tool (`dsconfig`) commands necessary to configure your server instance. For more information on using batch files, see Using dsconfig 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 PingDirectory 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 PingDirectory 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
```

Directory server folder layout

After extracting the PingDirectory server's distribution file, you see the following folders and command-line utilities.

Layout of the PingDirectory server Folders

Directories/Files/Tools	Description
`License.txt`	Licensing agreement for the PingDirectory server.
`README`	Describes the steps to set up and start the PingDirectory server.
`bak`	Stores the physical backup files used with the **`backup`** command-line tool.
`bat`	Stores Windows-based command-line tools for the PingDirectory server.
`bin`	Stores UNIX/Linux-based command-line tools for the PingDirectory server.
`classes`	Stores any external classes for server extensions.
`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 PingDirectory server.

Directories/Files/Tools	Description
`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 PingDirectory 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 PingDirectory server.
`resource`	Stores the MIB files for SNMP and can include ldif files, make-ldif templates, schema files, `dsconfig` batch files, and other items for configuring or managing the server.
`revert-update`	The **`revert-update`** tool for UNIX/Linux systems.
`revert-update.bat`	The **`revert-update`** tool for Windows systems.
`setup`	The **`setup`** tool for UNIX/Linux systems.
`setup.bat`	The **`setup`** tool for Windows systems.
`scim-data-tmp`	Used to create temporary files containing 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.

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 backend serves the global domain data that resides above the entry-balancing point.
- One backend 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, you can manage the data existing with each backend independently. 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:
    1. Perform a **`dsreplication enable`** operation.
    2. 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.

Generating sample data

The 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 the PingDirectory server with a range of dataset sizes. The templates are located in `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.

  Example:

```
$ 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.

Result:

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 PingDirectory 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 the PingDirectory server, see Managing Entries.
2. Stop the PingDirectory server.
3. To import data from an LDIF file to the PingDirectory 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 because of 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.

   Example:

   ```
   $ bin/import-ldif --backendID userRoot --ldifFile /path/to/data.ldif --rejectFile
       rejects.ldif --skipFile skipped.ldif --overwrite --overwriteExistingEntries --
   stripTrailingSpaces
   ```

4. Restart the PingDirectory server.

Running the server as a Microsoft Windows service

The server can run as a Windows service, which enables you to sign 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.

> **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.

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

1. These configurations can be changed in the `<server-root>\config\wrapper.conf` file.

Running the status tool

The PingDirectory 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 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.

   Example:

```
$ 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
-------------:----------:---------
Step 0.0.0.0:1389 : LDAP     : Enabled
Step 0.0.0.0:1689 : JMX      : Disabled
Step 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

The 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 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 the server and its tools.

About minimizing disk access

Minimizing disk access is critical to the PingDirectory server's 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 PingDirectory 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 PingDirectory server footprint and logic behind the automated tuning of the **`setup`** tool.

The PingDirectory 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

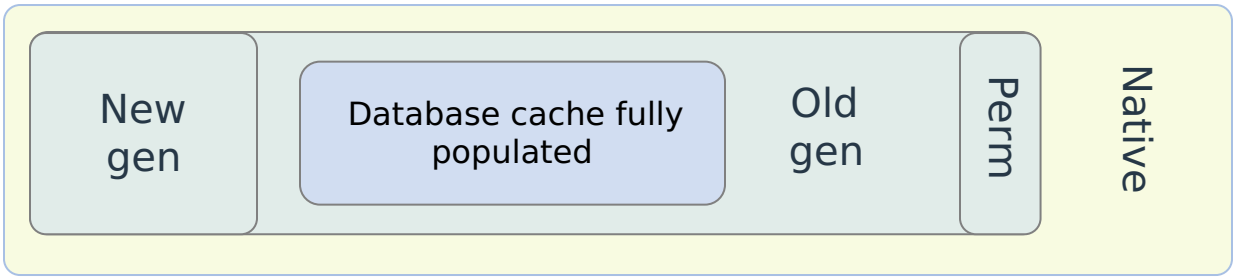
The **`setup`** tool for the PingDirectory server automatically assigns the JVM options and determines the memory allocation based on the total amount of memory on the system; however, in most production deployments, additional tuning might be required to meet the performance objectives for your system.

Often 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.

PingDirectory server process memory

The PingDirectory 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.

Although the JVM heap is referred to frequently as the maximum memory consumed by the PingDirectory server, the actual process size is slightly larger than the `Xmx` value because of the accumulation of small chunks of native code that Java requires for items such as SSL sockets.



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, with frequent garbage collection. Any data that's 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's size isn't explicitly stated in the JVM options; it's computed from the leftover heap after defining the `MaxHeapSize` and new generation sizes.

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
```

Important

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 **`setup`** and **`dsjavaproperties`** tools set `MaxNewSize` and `NewSize` values based on the results of extensive performance testing and shouldn't need to be changed.

Determining heap and database cache size

Define proper memory allocation of the PingDirectory server components using the `setup` command.

About this task

To define the proper memory allocation of the PingDirectory server's components:

Steps

1. Run the `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.

   Example:

   Set the `prime-method` to `preload` on the `userRoot` backend configuration.

4. Restart the PingDirectory server and watch for a successful preload message.

   Choose from:

   - 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:

   1. 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.
   2. Run the following command.

      ```
      bin/dsjavaproperties
      ```

   3. 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.

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.

Automatic DB cache percentages

Adjust the allocation of the DB cache percentage when setting up the PingDirectory 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 PingDirectory server allocates a maximum JVM heap of 384 MB.

About this task

The 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:

  Choose from:

  - 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 PingDirectory 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 PingDirectory server's JVM.

`.Offline`

Can require the same amount of memory needed by the 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 PingDirectory server's 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 PingDirectory 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 PingDirectory 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 PingDirectory server. The following procedures show how to configure database preloading.

Configuring database preloading

You can configure database preloading to load the database contents from disk into memory when the PingDirectory server starts up.

Steps

1. Configure database preloading.

   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.

   Example:

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

2. To apply the changes, restart the PingDirectory server.

   1. Run `bin/stop-server`.
   2. Run `bin/start-server`.

Configuring multiple preloading methods

You can configure multiple preloading methods to achieve the benefits of preloading without delaying PingDirectory server startup.

Steps

1. 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 after the server has been started with the `cursor-across-indexes` option.

   Example:

   ```
   $ 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 PingDirectory server.

   1. Run `bin/stop-server`.
   2. Run `bin/start-server`.

Configuring system index preloading

You can configure system index preloading to reduce the PingDirectory server's startup time.

About this task

Some environments have many indexes configured though only a few are used for performance-sensitive traffic.

Steps

1. Preload only the necessary indexes into the database at startup.

   Example:

   ```
   $ 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 PingDirectory server.
    1. Run **`bin/stop-server`**.
    2. Run **`bin/start-server`**.

Databases on storage area networks, network-attached storage, or running in virtualized environments

Several considerations exist 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. A full backup strategy 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 PingDirectory 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 because of 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 PingDirectory 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

The 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.

By default, PingDirectory generates tokens for the following:

- Base DNs for the backend, such as `dc=example,dc=com`
- Any `compact-common-parent-dn` values defined in the server configuration
- `ou=People,<base dn>`
- `ou=Groups,<base dn>`

The tokens are generated in this order. You can also define additional common base DNs to be tokenized.

Steps

- Use the following configuration to tokenize two branches: `ou=people,dc=example,dc=com` and `ou=customers,dc=example,dc=com`.

  Example:

```
$ 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.

  Example:

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

JVM properties for server and command-line tools

The PingDirectory 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 PingDirectory server and each command-line utility includes the default *<JAVA_HOME>* path.

The `java.properties` file 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 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** utility 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 PingDirectory 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 PingDirectory server compares the MD5 digest with the version stored in **set-java-home** and sends the following message to standard error if the digests differ:

```
WARNING -- File /ds/{pingdir}/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 PingDirectory 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.

  Example:

  ```
  $ 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 PingDirectory 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.

  Example:

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

Tuning for disk-bound deployments

You can configure the PingDirectory server for a disk-bound configuration.

About this task

For best performance, configure the server to fully cache the DIT in the backend database cache. The 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 PingDirectoryProxy 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 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 PingDirectory 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 because of hardware or financial constraints, or in which acceptable performance can be achieved by only caching a portion of the data.

The PingDirectory 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 PingDirectory 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 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 PingDirectory 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.

   Example:

   ```
   $ 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.

   Example:

   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.

   Example:

   ```
   $ 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.

   Example:

   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 ZGC

The Z Garbage Collector (ZGC) is a low-latency garbage collector that is well-suited for large Java Virtual Machine (JVM) heaps. PingDirectory supports both the non-generational and generational types of ZGC.

> **Note**
>
> For the rest of this article, "ZGC" indicates the non-generational type of garbage collection and "Generational ZGC" indicates the generational type.

Supported JDK versions

Of the supported JDK versions for PingDirectory, ZGC requires at least JDK 17 (for full support) and Generational ZGC requires at least JDK 21. Learn more about the JDK version requirements for ZGC in the OpenJDK Wiki.

Heap space recommendations

> **Note**
>
> These recommendations apply to both ZGC and Generational ZGC.

In the JVM startup arguments, you should configure the heap space with at least 20 - 25% headroom space left for the system RAM. For example, for a server with 62 GB RAM, the `xmx` and `xms` arguments in the `start-server.java-args` argument of the `java.properties` file should be set to 45 GB, expressed by `-Xmx45g -Xms45g`.

ZGC

ZGC is a single-generation, region-based, NUMA-aware, compacting collector. Like G1, ZGC works concurrently with the JVM application. ZGC doesn't track the number of collection cycles that objects in the heap have survived. Because each region of the heap is concurrently scanned in each cycle, objects aren't sorted into new or old generations. This collector is suitable for applications with large amounts of memory that require short pause times, as it does not pause the execution of application threads for more than 10 ms.

The recommended JVM properties when using ZGC for garbage collection are:

```
-XX:+UseZGC -XX:ConcGCThreads=10 -XX:InitiatingHeapOccupancyPercent=80 -XX:MaxNewSize=2g -XX:NewSize=2g
```

Generational ZGC

In contrast to ZGC, Generational ZGC *does* separate the heap into new and old generations to optimize the collection process.

The recommended JVM properties when using Generational ZGC for garbage collection are:

```
-XX:+UseZGC -XX:+ZGenerational -XX:ConcGCThreads=10 -XX:InitiatingHeapOccupancyPercent=80 -XX:MaxNewSize=2g
-XX:NewSize=2g
```

Learn more about Generational ZGC in HotSpot Virtual Machine Garbage Collection Tuning Guide in the Oracle Java documentation.

Changing the JVM garbage collector type

You can change the JVM garbage collector type by using a CLI tool or by editing the Java properties file manually.

Before you begin

> **Caution**
>
> Before attempting to change the JVM garbage collector type:
>
> - Consult Ping Identity Support.
> - Confirm that your environment meets the JDK version requirements for your targeted garbage collection type.

If you proceed, ensure that you have a backup copy of your `java.properties` file in the `config` directory. The backup file usually has the same file name with a `.old` extension. If you don't have a backup, do one of the following:

- Run **`bin/dsjavaproperties --initialize`**.

  > **Important**
  >
  > Each time you run this command, the server overwrites the `java.properties.old` file if it exists. It can also overwrite any changes you made to `java.properties` after installing the server.

- Copy the file to a safe directory manually.

About this task

The server supports the following garbage collection types:

Name	Encoded name	Supported JDK version(s)
G1	`g1`	11, 17, 21
ZGC	`zgc`	17, 21
Generational ZGC	`generational-zgc`	21
CMS	`cms`	8, 11

Steps

- Change the garbage collector type by doing one of the following.

> **Important**
>
> These choices involve running **`$ bin/dsjavaproperties --initialize`**, which can overwrite any changes you made to `java.properties` after installing the server.

Choose from:

- When making changes to multiple servers, use the **`dsjavaproperties`** tool.
  - Run the following command, substituting the encoded name of the garbage collector from the previous table:

```
$ bin/dsjavaproperties --gcType <encoded_gc_name> --initialize
```

- When making changes to one server, edit the `java.properties` file manually.
  - Change the value of the `gc-type` parameter to the encoded name of the garbage collector from the previous table.
  - Save the `java.properties` file.
  - Run **`$ bin/dsjavaproperties --initialize`**.

- Restart the server.

Troubleshooting

If something goes wrong and you need to restore your `java.properties` file to the backup version, do the following:

1. Add a `.bad` extension to the malfunctioning `java.properties` file.
2. Change the name of the working backup file to `java.properties` and, if needed, place it in the `config` directory.
3. Run **`$ bin/dsjavaproperties`** with no arguments.
4. Restart the server.

Configuring the PingDirectory server

Configure the PingDirectory 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 server applications. However, additional tuning might be necessary to meet the needs of your production environment.

The PingDirectory server stores its configuration settings in an LDIF file, `config/config.ldif`. Rather than editing the file directly, the server provides command-line options and an admin console for configuration purposes. The PingDirectory server also includes tools to create server groups to apply configuration changes to multiple servers at one time.

About the configuration tools

You can access and modify the server configuration in two ways.

Admin console

The server provides an admin console for graphical server management and monitoring. The console functions are equivalent to the **`dsconfig`** tool for viewing or editing configurations.

All configuration changes using the admin console are recorded in `logs/config-audit.log`, which also has the equivalent reversion commands if you need to undo a configuration.

`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
- 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 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.

To view the options for the dsconfig tool, change to the `PingDirectory/bin` directory, and enter `./dsconfig --help`. Example output is shown below.

```
./dsconfig --help

View and edit the Directory Server configuration.
This utility offers three primary modes of operation, the interactive mode, the non-interactive mode
and batch mode.  The interactive mode supports viewing and editing the configuration via an intuitive,
menu driven environment.  Running dsconfig in interactive command-line mode provides a user-friendly,
menu-driven interface for accessing and configuring the server. To start dsconfig in interactive
command-line mode, simply invoke the dsconfig shell script or batch file without any arguments.

The dsconfig non-interactive command-line mode provides a simple way to make arbitrary changes to the
Ping Identity Directory Server by invoking it on the command-line. If you want to use administrative
scripts to automate the configuration process, then run the dsconfig command in non-interactive mode.

The dsconfig tool provides a batching mechanism that reads multiple dsconfig invocations from a file
and executes them sequentially.  The batch file provides advantages over standard scripting in that it
minimizes LDAP connections and JVM invocations that normally occur with each dsconfig call.  You can
view the logs/config-audit.log file to review the configuration changes made to the Ping Identity
Directory Server and use them in the batch file.

Subcommands

  See the Usage section for instructions on viewing the list of supported subcommands.

Usage:  dsconfig  {options}
        where {options} include:

    --applyChangeTo [server-group|server-group-force|single-server]
        Controls whether changes apply to a single server or all servers in the configuration server group
    --offline
        Interact with the local configuration while the server is offline.  Not for use while the server
        is running
    -r, --reason {reason}
        A string describing the reason for the configuration change
    --help-classifications
        Display subcommands relating to connection and operation classification
    --help-core-server
        Display subcommands relating to core
    --help-database
        Display subcommands relating to backends, indexing, and caching
    --help-logging
        Display subcommands relating to logging, monitoring, and notifications
    --help-replication
        Display subcommands relating to replication
    --help-security
        Display subcommands relating to security and authorization
    --help-topology
        Display subcommands relating to topology
    --help-user-management
        Display subcommands relating to authentication and password management
    --help-web
        Display subcommands relating to web services and applications
    --help-subcommands
        Display all subcommands
```

```
  Configuration Options

    --advanced
        Allow the configuration of advanced components and properties

  LDAP Connection Options

    -Z, --useSSL
        Use SSL for secure communication with the server
    -q, --useStartTLS
        Use StartTLS to secure communication with the server
    --useNoSecurity
        Use no security when communicating with the server
    -h, --hostname {host}  [Default: localhost]
        Directory Server hostname or IP address
    -p, --port {port}  [Default: 389]
        Directory Server port number
    -D, --bindDN {bindDN}  [Default: cn=Directory Manager]
        DN used to bind to the server
    -w, --bindPassword {bindPassword}
        Password used to bind to the server
    -j, --bindPasswordFile {bindPasswordFile}
        Bind password file
    -o, --saslOption {name=value}
        SASL bind options (can be specified multiple times)
    -X, --trustAll
        Trust all server SSL certificates
    -P, --trustStorePath {truststorePath}  [Default: /Users/rowannabobo/Desktop/PingDirectory_9.2/config/
truststore]
        Certificate truststore path
    -T, --trustStorePassword {truststorePassword}
        Certificate truststore PIN
    -U, --trustStorePasswordFile {path}
        Certificate truststore PIN file
    --trustStoreFormat {trustStoreFormat}
        Certificate truststore format
    -K, --keyStorePath {keystorePath}
        Certificate keystore path
    -W, --keyStorePassword {keystorePassword}
        Certificate keystore PIN
    -u, --keyStorePasswordFile {keystorePasswordFile}
        Certificate keystore PIN file
    --keyStoreFormat {keyStoreFormat}
        Certificate keystore format
    -N, --certNickname {nickname}
        Nickname of the certificate for SSL client authentication

  Utility Input/Output Options

    -v, --verbose
        Use verbose mode
    -Q, --quiet
        Use quiet mode
    -n, --no-prompt
        Use non-interactive mode.  If data in the command is missing, you will not be prompted and the
```

```
        tool will fail
    -F, --batch-file {batchFilePath}
        Path to a file containing a sequence of dsconfig commands to run
    --batch-continue-on-error
        Force the execution of all commands in the batch file on the server even if prevalidation fails.
        Execution will also continue even if one of the commands fails.
        Please note that commands affecting multiple servers can still fail to execute unless the
        --applyChangeTo argument is provided with the value server-group-force. Only applies if the batch
        file argument is also supplied.
    --dry-run
        Validate configuration changes but do not apply them. This option can only be used along with the
        -F/--batch-file option
    --propertiesFilePath {propertiesFilePath}
        Path to the file that contains default property values used for command-line arguments
    --noPropertiesFile
        Specify that no properties file will be used to get default command-line argument values
    --script-friendly
        Use script-friendly mode

  General Options

    -V, --version
        Display Directory Server version information
    -?, -H, --help
        Display general usage information
    --help-ldap
        Display help for using LDAP options
    --help-sasl
        Display help for using SASL options
    --help-debug
        Display help for using debug options

Examples

  Start dsconfig in interactive mode:

    dsconfig

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

    dsconfig --no-prompt --bindDN uid=admin,dc=example,dc=com \
         --bindPassword password 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

  Use batch mode to read and execute a series of commands in a batch file:

    dsconfig --bindDN uid=admin,dc=example,dc=com --bindPassword password \
         --no-prompt --batch-file /path/to/config-batch.txt

  List information about all available configuration properties for all objects, including inherited
properties:
```

```
    dsconfig list-properties --offline --inherited

  For examples and help with LDAP options see --help-ldap.  For help with SASL authentication, see --help-
sasl
```

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.

   Result:

   You are prompted for connection and authentication information to the PingDirectoryProxy 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 PingDirectory 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.

   Example:

   ```
   $ bin/dsconfig
   ```

2. Enter the LDAP connection parameters.

Choose from:

- Enter the PingDirectory server's 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 PingDirectory 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 PingDirectory server using dsconfig interactive mode.
2. To display the advanced properties, from the **Local DB Backend properties** menu, enter `a`.

   Result:

   This toggles any hidden properties.

Changing the dsconfig object menu

The purpose of object levels is to present 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 Installing the PingDirectory server 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.

   Choose from:

   - 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 server components are displayed.

dsconfig interactive administrative alerts

The `dsconfig` tool and the admin 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]: One or more configuration property changes require administrative action or confirmation/
notification. Those properties include: * db-directory: Modification requires that the PingDirectory server be
stopped, the database directory manually relocated, and then the PingDirectory server restarted. While the
PingDirectory 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 admin console. For more information on the properties, see the PingDirectory Server Configuration Reference, located in the server's `do cs/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 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.

   Example:

   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 PingDirectory 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.

  Example:

```
$ 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.

   Example:

```
$ 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.

   Example:

```
$ 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.

   Example:

   ```
   $ 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 PingDirectory server the way you want and then get the **`dsconfig`** arguments from the log. Another way is to configure an option using the admin console. The console shows the equivalent **`dsconfig`** command before 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 enter the letter `f` to apply the changes.
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 message.

```
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 the command is presumed to be used to connect to a different remote host.

Using dsconfig batch mode

Configure the server in `dsconfig` batch mode.

About this task

The PingDirectory server provides a **`dsconfig`** batching mechanism that reads multiple **`dsconfig`** invocations from a file and executes them sequentially.

The batch file provides advantages over standard scripting by minimizing LDAP connections and 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 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 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 server also provides a `docs/sun-ds-compatibility.dsconfig` file for migrations from Oracle to PingDirectory 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 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.

   Example:

```
# 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 the PingDirectory server or the 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 the PingDirectory server or the 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.

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 admin console to create recurring tasks and task chains to perform regular maintenance tasks for the 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 recurring tasks and 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.

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.
> You cannot 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`.

   Example:

   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`.

   Example:

```
$ 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.

  Example:

  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 primary and secondary architecture for mirroring shared data across the topology.

All writes and updates are forwarded to the primary, 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 primary requirements and selection

A topology primary 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 primary sends a digest of its subtree contents. If the node's digest differs from the primary's, the server node knows it is not synchronized. The servers pull the entire subtree from the primary if they detect that they are not synchronized.

A server detects it is not synchronized with the primary under the following conditions:

- The server's subtree digest differs from the primary's digest at the end of its periodic polling interval.
- One or more servers have been added to or removed from the topology.

The primary 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 primary, the topology data is reviewed from all available servers, every five seconds by default, to determine if any new information identifies a better server primary. If a new server can be the primary and no other servers have indicated that they should be the primary, it will communicate its eligibility to the other servers. This ensures that all servers accept the same primary at approximately the same time, within a few milliseconds of each other. If there is no better primary, the initial primary maintains the role.

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

- A majority of servers is reachable from that primary.

  The primary server itself is considered while determining this majority.

- There is only a single primary in the entire topology.

If either of these conditions is not met, the topology is without a primary and the peer polling frequency is reduced to 100 milliseconds to find a new primary as quickly as possible. If there is no primary 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 primary 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 primaries, 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 primary because most of the servers are not available. A server with this setting enabled is assigned the role of primary if no suitable primary 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 primary 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 primary is found, the polling frequency adjusts to 100 milliseconds until a new primary 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 primary on all of the servers in the topology. The default is 10 seconds, but updates can take up to twice as long if primary 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 primary, if the primary is not known
- The instance name of the current primary or if a primary is not set, a description stating why a primary is not set
- A flag indicating which server thinks that it should be the primary
- A flag indicating which server is the current primary
- A flag indicating a server that was forced as primary
- 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 primary or another node in the topology if the primary is unknown
- The amount of time in milliseconds that multiple primaries were detected by this server
- The amount of time in milliseconds that no suitable server is found to act as primary
- 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 primary:

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

Configure passphrase providers

Passphrase providers allow access to clear-text passphrases, application programming interface (API) keys, and other secrets through an extensible API that are needed for essential processing tasks.

These secrets can be used to:

- Access external services. For example, the PingDirectoryProxy server can use a clear-text secret for accessing the credentials needed for authentication to backend directories, and the PingDataSync server can use a secret for authentication to the synchronization source and destination servers.
- Access certificate key and trust stores.
- Reversibly encrypt passwords stored in the LDAP changelog. This allows PingDataSync to decrypt these passwords and send the clear-text value to the destination server.

The PingDirectory server supports the following passphrase providers:

Amazon Secrets Manager passphrase provider

Reads a passphrase from the Amazon AWS Secrets Manager service. The provider can only be used with string secrets, in which the Secrets Manager service returns the secret in the form of a JavaScript Object Notation (JSON) object, and not with secrets stored in binary form.

Azure Key Vault passphrase provider

Reads a passphrase from the Microsoft Azure Key Vault service.

Conjur passphrase provider

Reads a passphrase from a CyberArk Conjur instance.

Environment Variable passphrase provider

Reads a passphrase from a specified environment variable.

File-based passphrase provider

Reads a passphrase from a specified file. The contents of the file can be encrypted with a key from the server's encryption settings database.

Obscured-value passphrase provider

Reads a passphrase that is stored directly in the server configuration.

> **Important**
>
> Although this passphrase is encrypted, the encryption uses a hard-coded key that an attacker can use to obtain the clear-text value. As a result, you should not depend solely on this provider's obfuscation for keeping the passphrase secret.

Third-party passphrase provider

Reads a passphrase using a third-party implementation of the UnboundID server SDK. This provider supports the read-only `extension-class` property for specifying the Java class that extends or implements the SDK, and the `extension-argument` property for customizing the provider's behavior.

Vault passphrase provider

Reads a passphrase from a HashiCorp Vault instance.

For example, to create an Amazon Secrets Manager passphrase provider, define the new passphrase provider in the server configuration:

```
dsconfig create-passphrase-provider \
  --provder-name "Amazon Secrets Manager" \
  --type amazon-secrets-manager \
  --set enabled:true \
  --set aws-external-server:[AWS_EXTERNAL_SERVER_DN] \
  --set secret-id:[AWS_SECRET_ID] \
  --set secret-field-name:[SECRET_JSON_FIELD]
```

After you've created a passphrase provider, you must update the global configuration of the components that use the provider. For example, if you want to use the passphrase provider to obtain the PIN needed for accessing a certificate key store, you must set the `key-store-pin-passphrase-provider` property in the key manager configuration.

Using the Configuration API

The 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.

Result:

The API is made available on the HTTPS Connection Handler's host:port in the `/config` context. Because of 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	Comparabledsconfig 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.backgroundReadLimit=0" }`	`$ dsconfig set-backend-prop --backend-name userRoot \ --add je-property:je.env.backgroundReadLimit=0`
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\"]" }`	`$ dsconfig set-backend-prop --backend-name userRoot \ --remove je-property:je.cleaner.adjustUtilization=false`

Operation	PATCH request	Comparabledsconfig modify-[object]options
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" : "id2childrenIndexEntryLimit" }`	`$ dsconfig set-backend-prop --backend-name userRoot \ --reset id2childrenIndexEntryLimit`

Example

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"
  } ]
}
```

Example

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`.

Example

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.

Example

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.

Example

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.

Example

```
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 because of 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.

Example

The following is an example encoded error message.

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

Generating a summary of configuration components

The `config-diff` tool generates 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 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`

  Example:

`$ 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

The PingDirectory 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 the PingDirectory server's implementation is that the root user's rights are granted through a set of privileges. This allows the PingDirectory 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`.

Using separate administrator accounts

Each administrator should have their own account with their own credentials tailored to the level of access needed.

The `setup` utility only creates a single root user. Some directory servers from other vendors only support a single root account so all administrators are required to share that account. However, this is a bad practice for several reasons, including:

- It's difficult to audit the actions of individual administrators. With separate accounts for each administrator, it's easy to see who did what.
- Credentials have to be shared across multiple administrators. Shared credentials are at a greater risk of being leaked or compromised. With separate accounts for each administrator, each has its own credentials that are not shared with anyone else.
- If the credentials for that single root account are leaked or compromised (or if someone with legitimate access to those credentials leaves their position and no longer needs that access), then those credentials need to be changed. The new credentials need to be communicated to all of the administrators sharing that account, which can be difficult to do securely. If the new credentials are not communicated in advance of the change, then administrators who have not received them won't be able to manage the server. If the new credentials are communicated in advance of the change, then the old, potentially compromised credentials remain in effect for longer than necessary.
- Using a single root account makes it more difficult to use strong authentication, such as a certificate or two-factor authentication, in a secure manner. With separate accounts for each administrator, it's easier to use strong authentication in a secure manner.
- If there is only a single root account, then that account must have a level of access that permits everything that any administrator might need to do, even if all administrators don't need the same level of access. With separate accounts for each administrator, each can be given precisely the level of access that they need.

It might also be beneficial to remove or disable the account for the initial root user created by `setup` once all of the administrators have their own accounts.

Using separate accounts is mentioned in the following `config/sample-dsconfig-batch-files/create-topology-admin-user.dsconfig` batch file, which provides an example for creating a topology admin user with various restrictions.

```
# Although setup automatically creates an initial root user account that can
# be used to manage the server, we strongly recommend creating a separate
# account for each administrator rather than using a single shared account.
# This offers several benefits, including:
#
# * It is easier to determine which administrator performed a given action.
# * There is no need to share credentials or coordinate password changes.
# * It is easier to use strong authentication options like certificates or
#   two-factor mechanisms.
# * You can customize the level of access that each administrator has.
#
# We also recommend creating topology admin user accounts rather than root
# user accounts.  If you create a root user account, then it is only created
# in that one instance.  If you create a topology admin user, then that
# account will be available in all instances in the topology.
#
# Because administrative accounts are very powerful, they are high-priority
# targets for attackers to try to compromise.  They should be required to have
# strong credentials, and you may wish to require that they use strong
# authentication mechanisms (see other dsconfig batch files for configuring
# password validators and other password policy features).  You may also want
# to give them usernames that are not likely to be guessed by an attacker,
# even if they know information about the individuals administering the
# service.  For example, you may not want to use their name in the DN or
# username, choosing instead something that is less predictable or even
# completely random like a UUID.
#
# Once each administrator has their own account, we recommend that you disable
# or remove the initial root user created during setup.  See the
# disable-or-remove-the-initial-root-user.dsconfig batch file for more
# information about that.

#
# NOTE:  Do not edit this file directly.  Changing this file could prevent it
# from being updated during a server upgrade.  If you want to alter the
# dsconfig commands below before applying the configuration changes, copy this
# file to another directory and edit that copy.
#

# Create a new topology administrator account.  In this example, the user will
# be able to read the configuration but will not be allowed to update it, and
# will be given the bypass-read-acl privilege rather than bypass-acl, which
# only ensures they will be able to read entries but they will not be
# permitted to update them unless permitted by the server's access control
# configuration.  They will also only be permitted to communicate with the
# server over a secure connection, and they will only be permitted to
# authenticate with the EXTERNAL or UNBOUNDID-TOTP mechanisms.
dsconfig create-topology-admin-user \
     --user-name "[USER_NAME]" \
     --set "password:[PASSWORD]" \
     --set "first-name:[FIRST_NAME]" \
     --set "last-name:[LAST_NAME]" \
     --set "user-id:[USER_NAME]" \
     --set inherit-default-root-privileges:true \
```

```
    --set privilege:bypass-read-acl \
    --set privilege:-bypass-acl \
    --set privilege:-config-write \
    --set search-result-entry-limit:0 \
    --set time-limit-seconds:0 \
    --set look-through-entry-limit:0 \
    --set idle-time-limit-seconds:0 \
    --set require-secure-authentication:true \
    --set require-secure-connections:true \
    --set "allowed-authentication-type:SASL EXTERNAL" \
    --set "allowed-authentication-type:SASL UNBOUNDID-TOTP"
```

Unpredictable identifiers for server administrators

You can prevent online password guessing attacks by using unpredictable identifiers for users.

If an attacker doesn't know the name of the account, then it's another obstacle to overcome before they can authenticate as them.

An entry's DN is the most common identifier used to authenticate, as it's required for simple binds and is often used for SASL binds. For regular users, you should name accounts with the entryUUID attribute. However, this isn't feasible for root users or topology administrators because the configuration framework requires these entries to use `cn` as the naming attribute. Further, many SASL mechanisms allow identifying users with a username, which is correlated to the associated entry using an identity mapper, so using an unpredictable DN might not be enough to sufficiently interfere with an attacker's ability to target a server administrator.

The best way to obscure identifiers for root users and topology administrators is to choose unpredictable values for the `cn` attribute in their accounts and not include any predictable alternate bind DN values for those accounts. Although it is be possible to use randomly generated `cn` values, it should be sufficient to use more memorable strings as long as they aren't something an attacker is likely to guess even if they know the identities of those administrators.

Secure communication for server administrators

Requiring secure communication for server administrators is recommended to prevent unencrypted communication.

While you should only permit secure access to the server, many environments might still allow access to unencrypted communication. Unencrypted communication is potentially dangerous for any kind of client because it could be observed and altered by a malicious third party, but it's especially dangerous for administrative accounts. Even if the server is configured to allow some clients to issue insecure requests, you should ensure that administrative accounts always use secure communication.

The best way to require that a user only be permitted to communicate with the server over a secure connection is to include the `ds-auth-require-secure-connection` operational attribute in that user's entry with a value of `true`. Although the password policy configuration includes `require-secure-authentication`, and `require-secure-password` changes properties, those properties can only guarantee the safety of authentication credentials. They don't guarantee secure communication for other operations processed by that user. This is because it's possible to authenticate in a manner that the server considers secure, without exposing credentials to an observer, without using a secure connection such as using a SASL mechanism like GSSAPI.

> **Note**
>
> CRAM-MD5 and DIGEST-MD5 also currently fall into this category, but for this purpose they are no longer considered secure.

Although it's possible to enable this by default for root users and topology administrators, you shouldn't do so at the risk of affecting backward compatibility with existing deployments that expect to be able to use insecure communication for these accounts. If the server is configured to only accept secure connections, or to reject insecure requests, then this setting is not as necessary.

Managing root user accounts

The directory 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.

  Example:

  ```
  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 the servers, 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 PingDirectory server contains a privilege subsystem that allows for a more fine-grained control of privilege assignments.

> **Note**
>
> Creating restricted root user accounts requires assigning privileges and necessary access controls for actions on specific data or backends. Access controls are determined by how the directory is configured and the structure of your data. See Managing access control 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.
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.

Privilege	Description
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 PingDirectory 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 can 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.

   Example:

   ```
   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.

   Example:

```
$ 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.

Example:

```
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.

Example:

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

5. To verify the additions, use the `ldapsearch` tool.

Example:

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.

Example:

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
```

Result:

```
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.

   Example:

   Create an `Admin Password Policy`, then add the password policy to the account.

```
$ bin/dsconfig create-password-policy \
  --policy-name "Admin Password Policy" \
  --set "description:Password policy for administrators" \
  --set password-attribute:userpassword \
  --set "default-password-storage-scheme:Salted 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.

   Example:

   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.

   Example:

   For this example, name the file `rootuser.txt`.

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

2. To change the root user's password, run `ldappasswordmodify`.

   Example:

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

3. Remove the text file.

   Example:

   ```
   $ 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 PingDirectory server.

> **Note**
>
> The example uses a static group using the `GroupOfUniqueNames` object class.

Steps

1. Create an LDIF file with an example administrator group.

   Example:

   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.

   Example:

   ```
   $ 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 PingDirectory server to the new administrator, create another LDIF file.

   Example:

   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.

   Example:

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

5. To verify the additions, use the `ldapsearch` tool.

   Example:

   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.

   Example:

   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
   ```

Result:

```
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`.

   Example:

   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`**.

   Example:

   ```
   $ 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`**.

   Example:

```
$ 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`.

   Example:

   ```
   $ 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`.

   Example:

   ```
   $ 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 PingDirectory server instances into the server group labeled "group-one".

Steps

1. Create a group called "group-one" using `dsconfig`.

   Example:

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

2. Add any PingDirectory 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.

   Example:

```
$ 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.

   Example:

```
$ 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 PingDirectory server in the group "server-group" by using the `--applyChangeTo server-group` option.

   Example:

```
$ 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 PingDirectory server instance.

   Example:

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

   Result:

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

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

   Example:

   ```
   $ 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 PingDirectory server instance.

   The first PingDirectory server instance should be disabled.

   Example:

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

   Result:

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

9. View the property on the second PingDirectory server instance.

   Repeat this step on the third PingDirectory server instance to verify that the property is still enabled on that server.

   Example:

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

   Result:

   ```
   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's 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.
`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 PingDirectory server.
`maximum-connection-duration`	Specifies the maximum length of time that a connection associated with the client connection policy can remain established to the PingDirectory 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 PingDirectory 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.

Property	Description
`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.
`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.

Property	Description
`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.**
`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.**

Property	Description
`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 PingDirectory server to examine too many entries over the course of processing the request.
`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 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 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
- How the policy is evaluated
- Configuring a client connection policy using the console
- Configuring a client connection policy using dsconfig
- Restricting server access based on client IP address

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 several 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 admin console.

Steps

1. In the admin 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 PingDirectory 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.
   1. To change the criteria, click View and Edit.
   2. To add new criteria, click Select New.
   3. Select the operations allowed for clients that are members of this connection group.
   4. To make operations available to clients, use the Add and Remove buttons.
   5. 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 PingDirectory server.

You can also choose to include and exclude specific base DNs using the appropriate fields.

9. Save your changes.

   Choose from:

   - 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 PingDirectory server main menu, enter `o`, and then enter the number for the Standard menu.
3. From the PingDirectory 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.

   1. If you want to make any further modifications, enter the number corresponding to the property.
   2. 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 PingDirectory 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 several 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 PingDirectory server.
- To configure the server using the `dsconfig` tool on the local host, specify the loopback address to 127.0.0.1.

  Example:

```
$ 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.

   Example:

   The following example uses the `dsconfig` tool in non-interactive mode. It allows only the PingDirectory 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.

   Example:

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

   Result:

   After you have run the command, access is denied to remote IP addresses. The PingDirectory server does not require a restart.

3. Add a remote IP range to the criteria.

   > **Note**
   >
   > For the following example, add `10.6.1.*`.

   Example:

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

   Result:

   Access from any remote servers is allowed. The PingDirectory server does not require a restart.

4. To restore default behavior, remove the criteria from the connection policy.

   Example:

> **Tip**
>
> Include the LDAP or LDAPS connection parameters, such as host name, port, bindDN, bindPassword, with the `dsconfig` command.

Example:

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

Result:

The PingDirectory server does not require a restart.

Automatically authenticating clients that have a secure communication channel

The PingDirectory 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.

  Example:

  ```
  $ 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 PingDirectory 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 server into lockdown mode.

  Result:

  Any client request to the PingDirectory server in lockdown mode receives an `Unavailable` response.

- To take the PingDirectory 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`.

  Example:

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

Leaving lockdown mode

Steps

- To leave lockdown mode, run `leave-lockdown-mode`.

  Example:

```
$ 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.

  Example:

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

Configuring maximum shutdown time

The PingDirectory 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 PingDirectory 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.

  Choose from:

  - 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 PingDirectory server but can be accessed on another server.

Use referrals when entries are located on another server. The PingDirectory server implements two types of referrals depending on the requirement:

Referral on Update plugin

The PingDirectory 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 PingDirectory servers where one server is a primary 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 PingDirectory server to point to the primary server. For example, if a client application sends an add request on the second PingDirectory server, the PingDirectory server responds with a referral that indicates any updates should be made on the primary server. All read requests on the read-only server process normally.

Smart referrals

The PingDirectory 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 PingDirectory server. If the 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 PingDirectory server.

`port`

Specifies the port number of the PingDirectory 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 PingDirectory 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 PingDirectory 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.

   Example:

   ```
   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.

   Example:

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

3. To verify the addition, search for a user.

   Example:

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

   Result:

   ```
   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.

  Example:

  ```
  $ 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.

  Example:

  ```
  $ 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 server for legacy systems that require it.

About this task

The read-only PingDirectory server participates in replication but cannot respond to any update requests from an external client. You can configure the PingDirectory server by setting the writability mode to internal-only, which makes the server operate in read-only mode. The Read-only mode PingDirectory server can process update operations from internal operations but reject any write requests from external clients. Because the PingDirectory server cannot accept write requests, you can configure the server to send a referral, which redirects a client's request to a primary 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 PingDirectory server using that extension will effectively respond to external update requests, even though the PingDirectory 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 PingDirectory server in read-only mode.

Steps

1. Install two replicating PingDirectory 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.

   Example:

   ```
   $ 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.

   Example:

   ```
   $ 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.

Example:

```
$ 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
```

Result:

```
[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 PingDirectory 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.

There is 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
- Configuring HTTP operation loggers
- Example HTTP log publishers
- Configuring HTTP connection handlers

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 SCIM 2.0 servlet extension configuration. 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.

Result

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.

  Example:

  ```
  $ 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 admin 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 PingDirectory 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.
  1. Specify the path and name for the active log file.
  2. 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`.

  Example:

  ```
  $ 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"
  ```

  Result:

  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.

Example:

```
$ 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
```

Result:

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. The server's `start-server` tool outputs 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 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`.

   Example:

   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`.

   Example:

   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.

   Example:

```
$ 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.

Example:

```
$ 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`.

Example:

```
$ 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`.

Example:

```
$ 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 PingDirectory 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`.

  Example:

  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
  ```

  Example:

  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.

  Example:

  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.

  Example:

```
$ 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 that also have LDAP-specific log messages. The response to the API call includes a `Correlation-Id` header with the value `ee919049-6710-4594-9c66-28b4ada4b127`.

```
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"
     ]
 }
```

The correlation ID can be used to search the HTTP trace log for matching log records, as in the following example.

```
$  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="{pingdir}  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="{pingdir}  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="{pingdir}  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, as in the following example.

```
$  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 the PingDirectory server to use an HTTP proxy server

Some organizations configure their network so that internal systems cannot directly access the Internet, but instead must use an HTTP proxy server to access external services.

The PingDirectory server now supports using an HTTP proxy server in conjunction with the following components:

- The Amazon Key Management Service cipher stream provider
- The Amazon Secrets Manager cipher stream provider
- The Amazon Secrets Manager passphrase provider
- The Amazon Secrets Manager password storage scheme
- The Azure Key Vault cipher stream provider
- The Azure Key Vault passphrase provider
- The Azure Key Vault password storage scheme
- The PingOne pass-through authentication plugin
- The Pwned Passwords password validator
- The Twilio alert handler
- The Twilio OTP delivery mechanism
- The UNBOUNDID-YUBIKEY-OTP SASL mechanism handler

Setting up the server to use an HTTP proxy server involves two steps, which are described in the following sections.

1. Creating an HTTP proxy external server in the configuration.
2. Configuring the appropriate component(s) to use the HTTP proxy server.

Creating an HTTP proxy external server

An HTTP proxy external server configuration object provides information about a proxy server that should be used. At present, we only support HTTP proxy servers (which can handle both unencrypted HTTP and encrypted HTTPS connections), with or without authentication.

HTTP proxy external server definitions support the following configuration properties:

`server-host-name`

The resolvable name or IP address of the HTTP proxy server to use. This is required.

`server-port`

The port on which the HTTP proxy server is listening for connections. This is required.

`basic-authentication-username`

The username to use if the proxy server requires authentication. This should be omitted if the proxy server does not require authentication.

`basic-authentication-passphrase-provider`

The passphrase provider to use to obtain the password to use if the proxy server requires authentication. This should be omitted if the proxy server does not require authentication.

For example, you can use a configuration change like the following to create an HTTP proxy external server that does not require authentication:

```
dsconfig create-external-server \
     --server-name "Example HTTP Proxy Server" \
     --type http-proxy \
     --set server-host-name:proxy.example.com \
     --set server-port:3128
```

Configuring server components to use the HTTP proxy external server

Merely defining an HTTP proxy external server in the configuration does not cause the server to use that proxy server for anything. Instead, you must indicate which components should use that proxy server.

This is necessary because you might only need to use an HTTP proxy server for certain components. For example, it might be necessary when accessing external web services but not for services on the internal private network.

All of the components that support the use of an HTTP proxy server offer an `http-proxy-external-server` configuration property whose value should be the name of the appropriate HTTP proxy external server definition in the configuration. For example, to update the Pwned Passwords password validator to use the HTTP proxy server defined in the "Example HTTP Proxy Server" configuration object, use a configuration change like the following:

```
dsconfig set-password-validator-prop \
     --validator-name "Pwned Passwords" \
     --set "http-proxy-external-server:Example HTTP Proxy Server"
```

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.

Global configuration property	Description
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 caused by unnecessary DNS lookups, follow these recommendations:
    1. Maintain a connection pool in the client app rather than opening new connections for each bind.
    2. Add appropriate records to DNS, including PTR records.
    3. Add `options timeout:1` or `options single-request` in the `/etc/resolv.conf` file.
    4. If IPv6 requests are causing issues, add `-Djava.net.preferIPv4Stack=true` to the `start-server.java-args` line in 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 directory 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 for the PingDirectory server and the PingDirectoryProxy server
- `ds-auth-allowed-address`: Restricting access through operational attributes in user entities for the PingDirectory server
- Connection Criteria: Restricting server access based on client IP address for the PingDirectory server and the PingDirectoryProxy server
- Connection Handlers: Restricting server access using the connection handlers for a configuration reference guide for all servers

Configuring traffic through a load balancer

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.

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.

When you set the `use-forwarded-headers` property and enable an HTTP connection handler to use `Forwarded` or `X-Forwarded-*` headers, many intermediate HTTP servers add information about the original request that would otherwise be lost.

If `use-forwarded-headers` is set to `true`, the server uses the client IP address and port information in the `Forwarded` or `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.

> **Note**
>
> If both the `Forwarded` and `X-Forwarded-*` headers are included in the request, the `Forwarded` header takes precedence. The `X-Forwarded-Prefix` header only overrides the context path for HTTP servlet extensions, not for web application extensions.

Configuring traffic through a load balancer using `dsconfig`

About this task

To configure the directory server to get traffic through a load balancer and to record the actual client's IP address:

Steps

1. Edit the HTTP or HTTPS connection handler object and set `use-forwarded-headers` to `true` by running `dsconfig`.

   Example:

   ```
   dsconfig set-connection-handler-prop \
       --handler-name "HTTPS Connection Handler" \
       --set use-forwarded-headers:true
   ```

2. To finalize the changes to the HTTP or HTTPS connection handler, use `dsconfig` to restart the connection handler.

Example:

```
dsconfig set-connection-handler-prop \
    --handler-name "HTTPS Connection Handler" \
    --set enabled:false

dsconfig set-connection-handler-prop \
    --handler-name "HTTPS Connection Handler" \
    --set enabled:true
```

3. To provide the `X-Forwarded-*` information to your load balancer, consult your provider's guide on configuring the load balancer settings.

Configuring traffic through a load balancer using the admin console

About this task

To configure the PingDirectory server to get traffic through a load balancer and to record the actual client's IP address:

Steps

1. On the PingDirectory admin console Configuration page, click Connection Handlers.
2. To edit your HTTP or HTTPS connection handler, in the Connection Handlers list, select the connection handler you want to edit.
3. To enable `Forwarded` headers, go to Use Forwarded Headers and select the Enabled check box.
4. Click Save.
5. To finalize the changes to the HTTP or HTTPS connection handler, use `dsconfig` to restart the connection handler.

   Example:

   ```
   dsconfig set-connection-handler-prop \
       --handler-name "HTTPS Connection Handler" \
       --set enabled:false

   dsconfig set-connection-handler-prop \
       --handler-name "HTTPS Connection Handler" \
       --set enabled:true
   ```

   > **Note**
   >
   > Because disabling the connection handler brings down the admin console, you must complete this step in the command line instead of the admin console.

6. To provide the `X-Forwarded-*` information to your load balancer, consult your provider's guide on configuring the load balancer settings.

Using the HAProxy PROXY protocol

When LDAP clients communicate through a software load balancer that supports the PROXY protocol, you can forward that client information from the software load balancer to the backend destination server.

> **Important**
>
> The PingDirectory, PingDirectoryProxy, and PingDataSync servers only support the PROXY protocol for inbound LDAP connections using either PROXY header version 1 or 2.
> The servers don't support using the PROXY protocol with outbound LDAP connections or other communication protocols, such as HTTP or JMX.

Considerations and limitations

To enable PROXY protocol support for a server, you must create and configure a PROXY-dedicated LDAP connection handler. Here's why:

- An individual LDAP connection handler doesn't support both PROXY and non-PROXY protocol connections.
- The server CLI tools, the admin console, and server-to-server communication don't support the PROXY protocol. They require at least one LDAP connection handler that isn't configured for PROXY protocol support.
- To help prevent the loss of access to administrative functions, given the previous limitations, the following restrictions apply when enabling PROXY protocol support:
  - You can only enable PROXY protocol support for an LDAP connection handler when you create it.
  - You can't enable or disable PROXY protocol support for an existing LDAP connection handler.
  - The LDAP connection handlers created during setup aren't configured with support for the PROXY protocol.

Using the PROXY protocol with TLS

You shouldn't configure the software load balancer as a TLS endpoint for LDAPS clients. Instead, you should pass the encrypted traffic through to the PingDirectory, PingDirectoryProxy, or PingDataSync server. This ensures the following:

- The communication from the client to the server remains end-to-end encrypted.
- The server can validate any presented client certificate chain.
- The server can use the certificate chain to authenticate the client by using the SASL EXTERNAL mechanism.

Using the PROXY protocol with PingDirectoryProxy

If you put the software load balancer in front of a PingDirectoryProxy server, you can use the PROXY protocol for communication between the end client and the PingDirectoryProxy server. The PingDirectoryProxy server then uses the intermediate client control to convey the end client address, which was included in the PROXY protocol header, to the backend PingDirectory server.

Enabling PROXY protocol support

Steps

To enable support for requests forwarded through a software load balancer that uses the PROXY protocol:

- Create a dedicated LDAP connection handler and set `use-haproxy-proxy-protocol` to `true`.

  Example:

  The following commands provide examples for creating LDAP and LDAPS connection handlers with support for the PROXY protocol, listening on ports 2389 and 2636, respectively:

```
$ bin/dsconfig create-connection-handler \
  --handler-name "LDAP Connection Handler With PROXY Protocol" \
  --type ldap \
  --set enabled:true \
  --set listen-port:2389 \
  --set allow-start-tls:true \
  --set ssl-cert-nickname:server-cert \
  --set key-manager-provider:JKS \
  --set trust-manager-provider:JKS \
  --set use-haproxy-proxy-protocol:true

$ bin/dsconfig create-connection-handler \
  --handler-name "LDAPS Connection Handler With PROXY Protocol" \
  --type ldap \
  --set enabled:true \
  --set listen-port:2636 \
  --set use-ssl:true \
  --set ssl-cert-nickname:server-cert \
  --set key-manager-provider:JKS \
  --set trust-manager-provider:JKS \
  --set use-haproxy-proxy-protocol:true
```

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 PingDataSync 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.

   Example:

```
$ 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 PingDirectory server. The plugin checks for uniqueness before 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 PingDirectoryProxy 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 PingDirectoryProxy server instead of enabling the attribute uniqueness plugin on the PingDirectory server. For more information, see the PingDirectory Server Administration Guide and the PingDirectoryProxy Server Administration 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`.

   Example:

   The following example checks for attribute uniqueness before an `ADD` operation using the `--set plugin-type:preoperationadd` option.

   ```
   $ bin/dsconfig set-plugin-prop --plugin-name "UID Unique Attribute" \
     --set enabled:true
   ```

   1. 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.

   Example:

   ```
   $ 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.

   Example:

   ```
   $ 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.

   Example:

   ```
   $ 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.

   Example:

   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.

   Example:

   ```
   $ bin/dsconfig create-json-attribute-constraints \
     --attribute-type sessioninfo \
     --set enabled:true
   ```

   Example:

   ```
   $ 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.

   Example:

   ```
   $ 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.

Before you begin
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.

Example:

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 pass-through authentication

The PingDirectory server provides support for passing through LDAP simple bind attempts to an external service for authentication processing, either instead of or in addition to the processing that it typically performs against the locally stored data.

Pass-through authentication can be useful when migrating to the PingDirectory server from a different type of datastore, especially when that datastore doesn't provide a means of directly migrating passwords.

The server provides pass-through authentication support for other LDAP servers (including Active Directory (AD), Oracle DSEE, OpenLDAP, and any other standards-compliant LDAPv3 server) and PingOne by default. You can also use the Server SDK to implement support for custom pass-through authentication handlers for interacting with other types of external services.

Configuration properties for pass-through authentication to LDAP servers

When used with the LDAP pass-through authentication handler, the pluggable pass-through authentication plugin can forward LDAP simple bind requests to another type of LDAP server for processing.

The following table contains the pluggable pass-through authentication plugin configuration properties.

Property	Description
`pass-through-authentication-handler`	The pass-through authentication handler that is used to interact with the external service. For passing through authentication to an LDAP directory server, create an LDAP pass-through authentication handler.
`included-local-entry-base-dn`	The base distinguished names (DNs) of subtrees containing local entries for which pass-through authentication is attempted. If this isn't provided, then all regular user entries (excluding root users and topology administrators) might be passed through.
`connection-criteria`	Optional connection criteria that can be used to indicate which clients can have their bind attempts passed through.
`request-criteria`	Optional request criteria that can be used to indicate which bind requests should be passed through.
`try-local-bind`	Indicates whether to try the bind attempt against the entry in the local server, only passing through to the external service if the local attempt fails. If this is `false`, then only pass-through authentication is used for applicable requests and local credentials aren't evaluated.

Property	Description
`override-local-password`	Indicates whether to pass through bind attempts for local accounts that have passwords. If this is set to `false`, bind attempts will only be passed through for accounts that don't have local passwords. This only applies if `try-local-bind` is `true`.
`update-local-password`	Indicates whether to update the password for the local account if authentication succeeds against the external service. This only applies if `try-local-bind` is `true`.
`update-local-password-dn`	The DN to use as the authorization identity when updating local passwords, which can be helpful if you want to synchronize other types of changes between the PingDirectory server and the external repository. If this isn't provided, an internal root account is used.
`allow-lax-pass-through-authentication-passwords`	Indicates whether to update the password for the local account even if it wouldn't have otherwise been accepted by the server (for example, if the password doesn't satisfy the configured set of password validators). This only applies if `update-local-password` is `true`.
`ignored-password-policy-state-error-condition`	Optionally allows pass-through authentication attempts to proceed against local accounts that are in certain states that don't allow them to authenticate locally (for example, if the account is locked or the password is expired).

The following table contains the LDAP pass-through authentication handler configuration properties.

Property	Description
`server`	The LDAP external servers to which bind attempts should be passed through.
`server-access-mode`	The mechanism that the server should use when choosing the order that the servers should be selected for pass-through authentication attempts.
`dn-map`	An optional mapping that can be used to construct the remote bind DN from the local PingDirectory server entry when authenticating to the external servers.
`bind-dn-pattern`	An optional pattern that can be used to construct the remote bind DN from the local PingDirectory server entry when authenticating to the external servers.

Property	Description
`search-base-dn`	The search base DN to use when searching for the corresponding entry in the external servers.
`search-filter-pattern`	An optional pattern you can use to construct a filter to search for the entry in the external servers that corresponds to an entry in the local PingDirectory server.
`initial-connections`	The initial number of connections to establish to each of the LDAP external servers.
`max-connections`	The maximum number of connections to maintain to each of the LDAP external servers.
`use-location`	Indicates whether to consider each server's location relative to the local PingDirectory server instance location when choosing the order that servers should be selected for pass-through authentication attempts.
`maximum-allowed-local-response-time`	The maximum length of time to wait for a response from an external server in the same location as the local PingDirectory server.
`maximum-allowed-nonlocal-response-time`	The maximum length of time to wait for a response from an external server in a different location from the local PingDirectory server.
`use-password-policy-control`	Indicates whether to include the password policy request control in bind requests forwarded to the external LDAP servers. This control can improve the server's ability to categorize authentication failures against the remote server, but not all types of LDAP servers support it. By default, the server includes the password policy request control if the server's root DSA-specific entry (DSE) advertises support for it.

At most one of the `dn-map`, `bind-dn-pattern`, and `search-filter-pattern` properties can be provided to indicate how the server should identify the entry in the remote server that corresponds to the entry in the local server. If none of these properties are provided, the local entry DN is used as the remote entry DN.

Configuring pass-through authentication to LDAP servers

To enable pass-through authentication to LDAP servers, create and configure at least one LDAP external server, a pass-through authentication handler, and a pluggable pass-through authentication plugin instance.

Steps

1. To create an LDAP external server, run `dsconfig create-external-server`.

   > **Note**
   >
   > If you already have an LDAP external server to use for pass-through authentication, proceed to step 2.

   Example:

   The following example creates two LDAP external servers.

   ```
   dsconfig create-external-server \
        --server-name ds1.example.com:636 \
        --type ldap \
        --set server-host-name:ds1.example.com \
        --set server-port:636 \
        --set connection-security:ssl \
        --set key-manager-provider:Null \
        --set trust-manager-provider:JKS \
        --set authentication-method:none

   dsconfig create-external-server \
        --server-name ds2.example.com:636 \
        --type ldap \
        --set server-host-name:ds2.example.com \
        --set server-port:636 \
        --set connection-security:ssl \
        --set key-manager-provider:Null \
        --set trust-manager-provider:JKS \
        --set authentication-method:none
   ```

2. To create an LDAP pass-through authentication handler, run `dsconfig create-pass-through-authentication-handler`.

   Example:

   ```
   dsconfig create-pass-through-authentication-handler \
        --handler-name LDAP \
        --type ldap \
        --set server:ds1.example.com:636 \
        --set server:ds2.example.com:636 \
        --set server-access-mode:round-robin
   ```

3. To create a pluggable pass-through authentication plugin instance, run `dsconfig create-plugin`.

   Example:

   ```
   dsconfig create-plugin \
        --plugin-name "Pluggable Pass-Through Authentication" \
        --type pluggable-pass-through-authentication \
        --set enabled:true \
        --set pass-through-authentication-handler:LDAP
   ```

The PingOne Pass-Through Authentication plugin

The PingOne Pass-Through Authentication plugin allows users to authenticate to the PingDirectory server with a password from PingOne and can optionally update the passwords in PingDirectory after successfully validating it in PingOne.

Although the PingDataSync server supports bidirectional synchronization between the PingDirectory server and PingOne, and it can synchronize password changes from PingDirectory to PingOne, it can't sync password changes from PingOne to PingDirectory. However, you can use the PingOne Pass-Through Authentication plugin to authenticate to the PingDirectory server with a PingOne password, and can optionally update the password in PingDirectory after successfully validating it in PingOne.

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 PingOne only if the local bind attempt fails.
- When `try-local-bind` is `false`, the plugin attempts to authenticate with PingOne first.

The following table identifies and describes the configuration properties associated with the PingOne Pass-Through Authentication plugin.

The PingOne Pass-Through Authentication plugin properties and their descriptions

Property	Description	Required	Default
`api-url`	URL that the PingDirectory server uses to communicate with PingOne.	Yes	N/A
`auth-url`	URL that the PingDirectory server uses to authenticate to PingOne.	Yes	N/A
`oauth-client-id`	OAuth client ID that the PingDirectory server uses to authenticate to PingOne.	Yes	N/A
`oauth-client-secret`	OAuth client secret that the PingDirectory server uses to authenticate to PingOne.	Yes	N/A
`environment-id`	Identifier for the PingOne 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 PingOne only for users in a specified distinguished name (DN). If this value is set, only users who exist within a specified base DN allow their authentication attempts to be passed through to PingOne.	No	All public naming contexts (if not set)

Property	Description	Required	Default
`connection-criteria`	Reference to a connection criteria object to use to identify the bind requests to pass-through to PingOne 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
`request-criteria`	Reference to a request criteria object to use to identify the bind requests to pass through to PingOne, 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 the PingDirectory server tries to process the bind locally before forwarding the bind request to PingOne. If this value is set to `true` and the bind succeeds locally, the PingDirectory server does not make a request to PingOne. If this value is set to `false`, the PingDirectory server ignores local credentials and attempts to authenticate only to PingOne.	Yes	`True`
`override-local-password`	Indicates whether the PingDirectory server attempts to bind to PingOne 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 PingOne only if the local user account doesn't have a password. If the local bind attempt fails while this value is set to `true`, the server tries to authenticate to PingOne even if the local account has a password.	Yes	`True`

Property	Description	Required	Default
`update-local-password`	Indicates whether the 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 PingOne 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 the PingDataSync server to synchronize password changes from the PingDirectory server into PingOne. If the passwords differ, either the local password or the password for PingOne allows the user to authenticate. If PingOne is the authoritative source for passwords, set this property to `true`. To ensure that a pass-through attempt to PingOne doesn't override local changes, make all password changes in PingOne.	Yes	`False`
`allow-lax-pass-through-authentication-passwords`	Indicates whether the PingDirectory server bypasses the normal password-validation process when setting the local password from PingOne. 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 PingOne 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 only if the authentication to PingOne succeeds, and if the password is accepted by the local password validators. If the PingOne password does not satisfy the configured set of password validators, the pass-through authentication attempt fails.	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 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 PingOne 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 field used to map local user entries to the corresponding PingOne 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 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 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.
`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.

Example

Configuring the PingOne Pass-Through Authentication plugin

To create and configure the PingOne Pass-Through Authentication plugin, run `dsconfig create-plugin` in a command similar to the following.

```
dsconfig create-plugin \
     --plugin-name "PingOne Pass-Through Authentication" \
     --type ping-one-pass-through-authentication \
     --set enabled:true \
     --set "api-url:<API URL>" \
     --set "auth-url:<Auth URL>" \
     --set "oauth-client-id:<Client ID>" \
     --set "oauth-client-secret:<Client Secret>" \
     --set "environment-id:<Environment ID>" \
     --set user-mapping-local-attribute:entryUUID \
     --set user-mapping-remote-json-field:externalId
```

Configuring pass-through authentication to custom services

The PingDirectory server provides support for passing through LDAP simple bind attempts to an external service for authentication processing, either instead of or in addition to the processing that it typically performs against the locally stored data. Use the Server SDK to implement support for custom pass-through authentication handlers for interacting with other types of external services.

About this task

To create a custom pass-through authentication handler, use the Server SDK. To configure your pass-through handler in the server:

Steps

1. Create an instance of a third-party pass-through authentication handler and set its `extension-class` property by running a command similar to the following.

   Example:

   ```
   dsconfig create-pass-through-authentication-handler \
        --handler-name "<Example Handler>" \
        --type third-party \
        --set extension-class:<com.example.ExamplePassThroughAuthenticationHandler>  \
        --set extension-argument:<argName1=argValue1>  \
        --set extension-argument:<argName2=argValue2>
   ```

   The third-party pass-through authentication handler supports the following configuration properties.

Property	Description
`extension-class`	The fully-qualified name of the Java class that provides the custom pass-through authentication handler implementation. This class must be a subclass of `com.unboundid.directory.sdk.ds.api.PassThroughAuthenticationHandler`.
`extension-argument`	An optional set of name-value pairs that provide arguments needed to configure the custom pass-through authentication handler.

2. (Optional) Set any further configuration needs, as determined by your custom implementation, through the `extension-argument` property.
3. After you have configured the third-party pass-through authentication handler, configure a pluggable pass-through authentication plugin instance to use it, using a command similar to the following.

   Example:

```
dsconfig create-plugin \
     --plugin-name "Pluggable Pass-Through Authentication" \
     --type pluggable-pass-through-authentication \
     --set enabled:true \
     -- set "pass-through-authentication-handler:<Example Handler>"
```

> **Note**
>
> For more information about the configuration properties for the pluggable pass-through authentication plugin, see Working with pass-through authentication.

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 PingDirectory server for Oracle compatibility

Configure the PingDirectory 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 PingDirectory 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 PingDirectory server's 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.

   Example:

   ```
   $ 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.

   Example:

   ```
   $ 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.

   Example:

   ```
   $ 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 PingDirectory 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.

   Example:

   ```
   $ 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.

   Example:

   ```
   $ 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

To sync passwords with PingOne, the PingDirectory server maps the `userPassword` attribute to the `password` attribute through a direct attribute mapping. (IBM Security Directory and the IBM Tivoli Directory servers return the `userPassword` attribute value as `userPassword;binary`.)

About this task

To sync passwords from thePingDirectory server to 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
```

The 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.

> **Note**
>
> RDBMS passwords cannot be encrypted and should be hashed with a scheme that PingDirectory server recognizes.

> **Important**
>
> The PingDataSync server cannot synchronize passwords between PingOne systems, because PingDataSync cannot retrieve passwords from PingOne.

Example:

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
```

Using the admin console

You can configure and maintain the PingDirectory server through the embedded admin console.

- Signing on to and configuring the admin console
- Single sign-on with the admin console
- Working with the schema
- Managing tasks
- Monitoring with the admin console

Signing on to and configuring the admin console

After you install the PingDirectory server, access the admin console to verify the configuration and manage the server.

About this task

The admin console is designed primarily for system configuration, which includes the creation of new schema elements like attribute types and object classes. Most entry-monitoring tasks, such as searching and modifying, must be executed with the built-in command-line tools. Learn more in Managing entries.

Steps

1. **Start the PingDirectory server.**

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

2. **To access the admin console, go to https://<*host*>:<*port*>/console/login.**

   <*host*> is the host name of the server, and <*port*> is the port on which the server accepts connections from HTTPS clients. The host name and port were configured during installation.

   > **Note**
   >
   > The `<port>` is different for PingDirectory and each of its add-ons. The admin console that you access is specific to the add-on for which you are using it.

3. **In the Username and Password fields, enter the root user DN credentials.**

   > **Note**
   >
   > The root user distinguished name (DN) or the common name of a root user DN is required to sign on to the admin console. For example, if the DN created when the server was installed is `cn=Directory Manager`, you can use `directory manager` to sign on.

4. **Click Sign In.**

   To set up an SSO connection, refer to Single sign-on with the admin console.

   To run the console in an external container, such as Tomcat, refer to Deploying the admin console.

Setting the admin console session timeout window

About this task

The default session timeout for the admin console is 24 hours. You can adjust your session length to suit your organization's needs.

> **Note**
>
> When the session duration is exceeded, all inactive users are logged off automatically.

To change the default session timeout value:

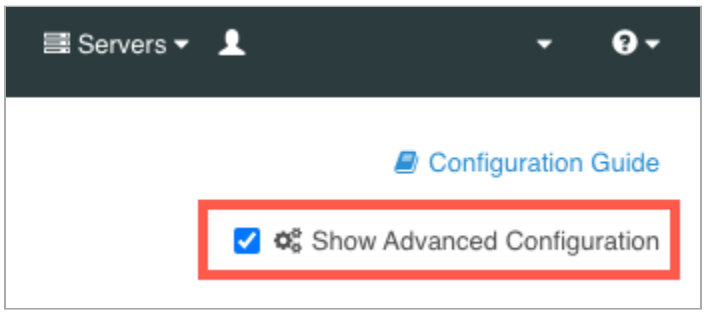
Steps

1. To configure the `server.sessionTimeout` application parameter, which specifies the timeout duration in seconds, set the value as an `init` parameter either in the console or on the command line:

   Choose from:

   - Use the admin console:

     1. Ensure that the Show Advanced Configuration check box is selected.

     2. In the Web Services and Applications list, select Web Application Extensions.
     3. In the Web Application Extension list, select Console.
     4. In the Init Parameter field, enter the desired timeout duration value in seconds.
     5. Click Save.

   - Use the command line to run the following:

     ```
     dsconfig set-web-application-extension-prop --no-prompt \
             --extension-name Console \
             --add init-parameter:server.sessionTimeout=<value in seconds>
     ```

2. To save your changes, restart the HTTP Connection Handler or the server:

   Choose from:

   - Restart the HTTP Connection Handler:

```
dsconfig set-connection-handler-prop \
    --handler-name "<HTTPS Connection Handler>" \
    --set enabled:false

dsconfig set-connection-handler-prop \
    --handler-name "<HTTPS Connection Handler>" \
    --set enabled:true
```

- Restart the server using `bin/stop-server` with the `-R` or `--restart` option:

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

Deploying the admin console

To administer multiple servers from a single console instance, you can use Docker or a Java application server, such as Tomcat or Jetty, to run the admin console as a standalone instance. Running the admin console as a standalone instance also allows you to make changes to the console without restarting the directory server's HTTPS connection handlers.

Learn more about deployment options in the following sections.

Using Docker to run a standalone admin console

About this task

The admin console is available as a standalone Docker image on Docker Hub. To see a `docker-compose` example that deploys a PingDirectory container with a corresponding console, see the PingDirectory getting started demo.

> **Note**
>
> The Ping Identity Helm charts also include support for the admin console, which you should use when deploying the console in Kubernetes.

Steps

- To connect with a PingDirectory server that is deployed in Docker, run the admin console image in a local Docker deployment.

  The network you use must match the network used by the PingDirectory server's container.

  Example:

  The following example deploys the standalone admin console image in a local Docker deployment with the console listening on port `8443` using the `pingnet` network:

```
docker run \
    --name pingdataconsole \
    --publish 8443:8443 \
    --network pingnet \
    --detach \
    pingidentity/pingdataconsole:edge
```

> **Important**
>
> You can change the first port in the `--publish` line based on what port you want to use on your local machine, but you shouldn't change the second port. To access the admin console outside of Docker on port `9443`, for example, use `9443:8443`.

- To access the admin console, go to https://*<hostname>*:*<port>*/console/login and enter the following credentials.

Field	Credential
Server	`<PingDirectory container name>:<LDAPS port>` **Note** The name of the server you enter on the sign-on page must match the name of the PingDirectory container you are trying to connect to along with the LDAPS port of that container. For example, use `pingdirectory:1636` to connect to a standalone PingDirectory Docker container on the same network.
Username	`administrator`
Password	`2FederateM0re`

Setting up the admin console on a Tomcat server

Before you begin

To set up a standalone admin console, you need the `.war` file from the PingDirectory server's `resource/admin-console.zip` archive.

About this task

You can use the `.war` file with Java application servers, such as Tomcat or Jetty. The following example uses Tomcat as the server environment.

Steps

1. Download the Tomcat 11 `.zip` archive from the Apache Tomcat downloads page.
2. Extract the `.zip` archive.
3. To set up the Tomcat server, follow the instructions in the `RUNNING.txt` file from the extracted directory.
4. To keep the Tomcat server information from being exposed, in the `Host` section of the `Tomcat_directory/conf/server.xml` file, add the following line:

```
<Valve className="org.apache.catalina.valves.ErrorReportValve" showReport="true"
showServerInfo="false" />
```

> **Note**
>
> This information is sourced from the Smart Scanner website, where you can learn more about this Tomcat server vulnerability.

5. Copy the `.war` file into the `webapps` folder in the Tomcat root directory.
6. To start the Tomcat server, run `startup.sh` or `startup.bat` from the `bin` folder of the Tomcat root directory.

   Result:

   Tomcat automatically extracts the admin console from the compressed `.war` file into an exploded application directory and starts running the console.
7. To access the admin console, add the name of the `.war` file to the end of the Tomcat server's path.

   Example:

   If the Tomcat server is deployed on `localhost:8080`, and the `.war` file is named `management-console.war`, then the path to access the console is `localhost:8080/management-console`.

Configuring the admin console

About this task

After you have deployed the PingDirectory admin console, you can configure it.

Steps

1. Disable the embedded admin console using `dsconfig` or the admin console to configure connection handlers:

   Choose from:

   - To use `dsconfig`, run `dsconfig set-connection-handler-prop`:

     ```
     dsconfig set-connection-handler-prop \
         --handler-name "<HTTPS Connection Handler>"  \
         --reset web-application-extension
     ```

     > **Note**
     >
     > Replace *<HTTPS Connection Handler>* with the name of the connection handler hosting the admin console.

   - To use the admin console, open the console:

     1. On the Configuration page, go to Connection Handlers.
     2. In the Connection Handlers list, select the HTTP or HTTPS connection handler that is hosting the admin console.
     3. Go to Web Application Extension and click the arrows to move Console from the Selected column on the right to the Available column on the left.
2. To finalize your changes, restart the HTTPS Connection Handler using `dsconfig`:

Example:

```
dsconfig set-connection-handler-prop \
    --handler-name "<HTTPS Connection Handler>" \
    --set enabled:false

dsconfig set-connection-handler-prop \
    --handler-name "<HTTPS Connection Handler>" \
    --set enabled:true
```

3. Configure the admin console's `application.yml` file.

   You can configure the standalone PingDirectory server admin console by modifying the `/tmp/Console/WEB-INF/classes/application.yml` file. To see the different configuration settings listed in the default `application.yml` file included with the admin console and what they do, expand the following table.

Setting	Description
`spring.*`	Learn more about these properties in the Spring API docs. You should not modify them.
`management.server.base-path`	Controls the prefix of the Spring Boot Actuator endpoints of the admin console application. You should not modify this setting.
`logging.level.*`	Controls the severity level of messages logged about these packages.
`log.console`	If this is set to `true`, the admin console logs messages to a file.
`log.file`	If logging is enabled, this specifies the file that the admin console will log to.
`PingData.SSO.OIDC.enabled`	If this is set to `true`, the admin console attempts to use OpenID Connect (OIDC) single sign-on (SSO) to bind to the managed server. If `false`, the admin console asks for a username and password.
`PingData.SSO.OIDC.issuer-uri`	The issuer URI to the OIDC provider.
`PingData.SSO.OIDC.client-id`	The client ID used with the OIDC provider.
`PingData.SSO.OIDC.client-secret`	The client secret used with the OIDC provider.
`PingData.SSO.OIDC.trust-store-file`	The file path to the trust store used when communicating with the OIDC provider.

Setting	Description
`PingData.SSO.OIDC.trust-store-type`	The type of trust store specified by `PingData.SSO.OIDC.trust-store-file`.
`PingData.SSO.OIDC.trust-store-pin`	Specifies the password used with the trust store specified by `PingData.SSO.OIDC.trust-store-file`.
`PingData.SSO.OIDC.trust-store-pin-environment-variable`	Specifies the environment variable containing the password used with the trust store specified by `PingData.SSO.OIDC.trust-store-file`.
`PingData.SSO.OIDC.strict-hostname-verification`	If this is set to `true`, the admin console requires a matching host name on the OIDC provider certificate.
`PingData.SSO.OIDC.trust-all`	If this is set to `true`, the admin console accepts any OIDC provider certificate.
`PingData.SSO.OIDC.username-attributes`	The LDAP attribute containing the username of the user the admin console is logging in as when using SSO.
`login.hide-server`	If this is set to `true`, the 'server' field is hidden on the sign on page.
`ldap.server`	Auto-populates the 'server' field on the sign-on page. If `login.hide-server=true`, this value determines which directory server the admin console tries to bind to.
`ldap.init-user`	Auto-populates the `user` field on the sign-on page.
`ldap.init-password`	Auto-populates the `password` field on the sign-on page.
`ldap.trust-store-file`	The file path to the trust store used when binding to the directory server.
`ldap.trust-store-type`	Specifies the type of trust store specified by `trust-store-file`.
`ldap.trust-store-pin`	Specifies the password used with the trust store specified by `trust-store-file`.
`ldap.trust-store-pin-environment-variable`	Specifies the environment variable containing the password used with the trust store specified by `trust-store-file`.
`ldap.file-servlet-name`	Specifies the name of the file servlet on the managed directory server to use when fetching generated `collect-support-data` (CSD) or server profiles.

Setting	Description
`ldap.csd-task-enabled`	If this is set to `true`, the admin console has a button that has the managed directory server run a `collect-support-data` task.
`ldap.csd-destination-folder`	The file path to the folder where the managed directory server stores generated CSD files after running the `collect-support-data` task.
`ldap.profile-destination-folder`	The file path to the folder where the managed directory server stores generated server profiles after running the `generate-server-profile` task. **Important** Do not change this property.
`branding.custom-folder`	The file path to the folder that holds custom `branding.properties`, `branding.css`, and `favicon.ico` files. If empty, default Ping Identity branding is used instead.
`configuration.complexity`	Determines the maximum complexity level for shown configuration objects. The possible values are `basic`, `standard`, `advanced`, and `expert`.
`server.sessionTimeout`	The amount of time a web session can remain idle before the user must sign on again. The time is set in seconds unless you use a time interval (h for hours or m for minutes). If not specified, the default is 24 hours.

Note

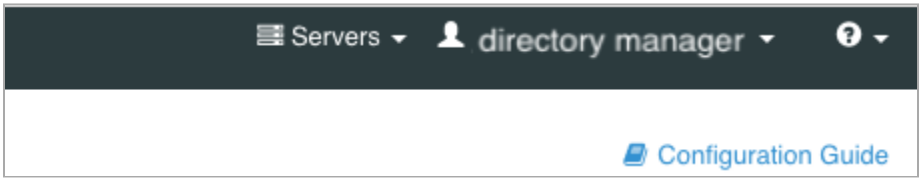
After modifying the `application.yml` file, you must restart the admin console for your changes to take effect.

4. Select servers to manage in the admin console:
    1. To use the `application.yml` file to select a server for the admin console to manage:
        1. Set the `ldap.server` property to the address of the LDAP server to bind to.
        2. Restart the console using the following command:

```
dsconfig set-connection-handler-prop \
    --handler-name "<HTTPS Connection Handler>" \
    --set enabled:false

dsconfig set-connection-handler-prop \
    --handler-name "<HTTPS Connection Handler>" \
    --set enabled:true
```

2. To switch between managed servers in a single topology while signed on to the admin console, in the Servers list, select the server that you want to manage.



3. To select a server when SSO is not enabled and the `login.hide-server` property in `application.yml` is `false`:
    1. If you are signed on to the admin console, sign off of your current session.
    2. Change the Server field value on the console sign-on page to the address of the LDAP server you want to manage.
4. To select a server when SSO is enabled, enter the admin console URL with the `ldap-hostname` and `ldaps-port` query parameters specified when accessing the console:

    `https://<hostname>:<port>/console/login?ldap-hostname=<ldap.host>&ldaps-port=<ldaps-port>`

    Example:

    In the following example URL, *<hostname>* is `localhost`, *<port>* is `443`, *<ldap.host>* is `ldap.host`, and *<ldaps-port>* is `636`:

    `https://localhost:443/console/login?ldap-hostname=ldap.host&ldaps-port=636`

Using custom rebranding

You can customize several visual elements of the admin console, including branding elements such as contact information or logos.

About this task

You can use the command line to configure the PingDirectory suite of products.

Steps

1. To customize the admin console, open the `application.yml` file and set *<branding.custom-folder>* to a local filepath that is not a subdirectory of the console application directory.

    This path should contain the elements for the admin console to display.

2. Set the custom branding filepath on any admin console in the PingDirectory suite of products by running the following `dsconfig` command:

```
dsconfig set-web-application-extension-prop \
    --extension-name Console \
    --set init-parameter:branding.custom-folder=<path to folder>
```

Customizing text information

About this task

To customize text information, such as contact information or company names:

Steps

1. Make a copy of the `branding.properties.template` file and rename it to `branding.properties`.
2. Make the desired changes to the copy.
3. Place the copy in the folder specified by the <*branding.custom-folder*> setting.
4. Restart the admin console.

Customizing the color scheme or logos

About this task

To customize the color scheme or logos used by the admin console:

Steps

1. Make a copy of the `branding.css.template` file and rename it to `branding.css`.
2. Make the desired changes.
3. Place the copy in the folder specified by the <*branding.custom-folder*> setting.

Customizing the page icon

About this task

To customize the page icon used by the admin console:

Steps

1. Name the desired icon `favicon.ico`.

> **Note**
>
> The icon must be in `.ico` format.

2. Place `favicon.ico` in the folder specified by the <*branding.custom-folder*> setting.

Single sign-on with the admin console

The OpenID Connect (OIDC) protocol enables single sign-on (SSO) with the PingDirectory server admin console using either PingOne, PingFederate, or a custom authorization server.

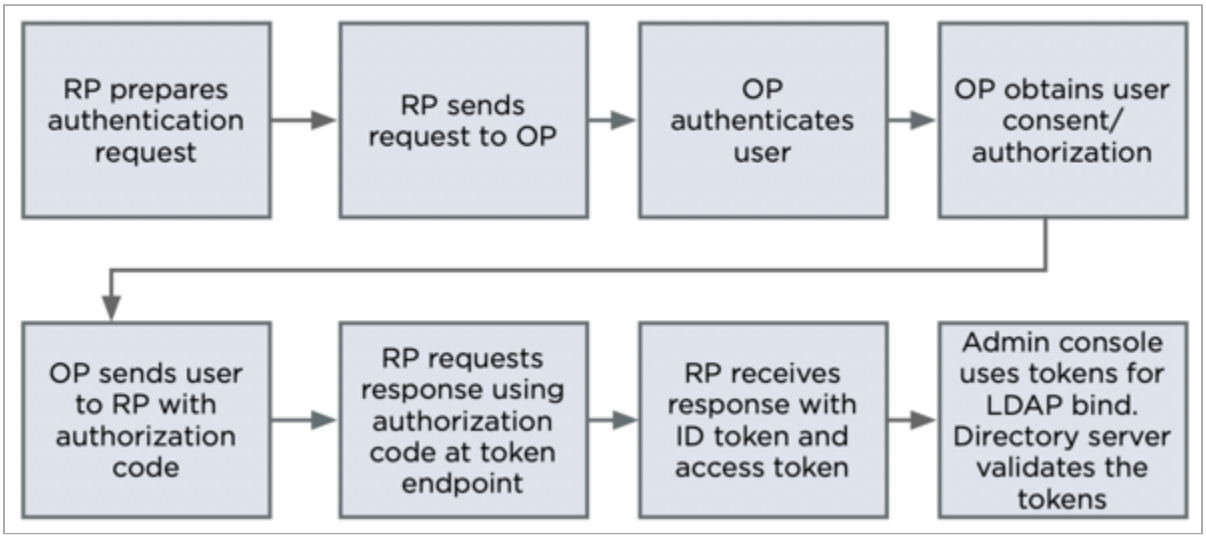
Overview of OpenID Connect

OIDC is a protocol that allows a client application called a relying party (RP) to confirm that a user is who they say they are by contacting an authorization server called an OpenID Provider (OP).

The authorization code flow

The PingDirectory server admin console uses the authorization code flow implementation of the protocol.

The following is the basic authorization code flow where the PingDirectory server admin console acts as the RP in steps 1 - 7, the managed server acts as the RP in step 8, and PingOne, PingFederate, or your custom authorization server acts as the OP.



1. The PingDirectory server admin console (RP) prepares an authentication request containing the desired request parameters.
2. The admin console (RP) sends the request to the authorization server.
3. The authorization server (OP) authenticates the user.
4. The authorization server (OP) obtains user consent/authorization.
5. The authorization server (OP) sends the user back to the admin console with an authorization code.
6. The admin console (RP) requests a response using the authorization code at the token endpoint.
7. The admin console (RP) receives a response that contains an ID token and access token in the response body.
8. The admin console attempts to use the tokens to perform an Lightweight Directory Access Protocol (LDAP) bind with the server (RP) through the OAUTHBEARER SASL mechanism. While establishing this LDAP bind, the server (RP) validates the tokens.

Configuring OIDC with the PingDirectory server admin console

The authorization server must be configured with the following settings.

Configuration requirement	Description
An OIDC client with a client ID, client secret, and a Redirection URI.	The client ID and client secret are used by the authorization server to confirm that the authentication and token requests are coming from a valid source. Set the redirection URI value to `https://<hostname>:<HTTPS_port>/console/oidc/cb`, where `https://<hostname>:<HTTPS_port>/console/` is where you access the console. **Note** The redirection URI value is used by the authorization server in steps 3 and 5 when sending the end user back to the RP. If the value is incorrect, the authorization server displays an error.
The authorization server must be able to return an ID token that maps to a user on the managed PingDirectory server.	The `sub` field in the ID token must be mappable.

The PingDirectory server admin console must be configured with the following settings in the `application.yml` file, which is located in `PingDirectory/tmp/console/webapp/WEB-INF/classes`.

Note

If doing this in DevOps, none of these settings will have any effect and the changes must be done through the profile.

application.ymlconfiguration name	Embedded console setting name	Settings required
`PingData.SSO.OIDC.enabled`	SSO Enabled	`true`. Necessary for the console to start the OIDC login flow. **Note** If SSO Enabled is set to `false`, the console asks for a username and password.
`PingData.SSO.OIDC.issuerUri`	OIDC Issuer URI	Allows the client to use OIDC discovery to determine the correct addresses to send the authentication request to in step 2 and the token request in step 6.

application.ymlconfiguration name	Embedded console setting name	Settings required
`PingData.SSO.OIDC.clientID`	OIDC Client ID	Value obtained from the authorization server. This is sent in the authentication request in step 2.
`PingData.SSO.OIDC.clientSecret`	OIDC Client Secret	Value obtained from the authorization server. This is sent in the token request in step 6.
`ldap.server`	LDAP Server	Set to the managed PingDirectory server's hostname and LDAPS port. Used in step 8 when the admin console attempts to perform an LDAP bind to the managed PingDirectory server using the OAUTHBEARER SASL mechanism.

The managed PingDirectory server must be configured with the ability to accept LDAPS connections and with the following additional configuration properties.

Configuration requirement	Description
A configured ID token validator	• For PingOne SSO: the PingOne ID token validator. • For PingFederate SSO or a general OIDC provider: the OIDC ID token validator. **Note** The ID token validator must be configured with an identity mapper that is able to map the `sub` value in the token to an LDAP user on the server.
A configured OAUTHBEARER SASL mechanism	Set the mechanism's ID token validator configuration property to the previously configured OIDC ID token validator. **Note** The OAUTHBEARER SASL mechanism is used in step 8 when the PingDirectory server processes the LDAP bind.

Setting up SSO to the PingDirectory admin console

- To set up SSO using PingOne, refer to Setting up SSO to PingDirectory from PingOne.
- To set up SSO using PingFederate, refer to Setting up SSO to PingDirectory from PingFederate.
- To set up SSO using a generic OP, refer to Setting up SSO to PingDirectory from a generic OpenID Connect provider.

Setting up SSO to PingDirectory from PingOne

To set up single sign-on (SSO) access from the PingOne admin console to the PingDirectory admin console, configure PingOne and PingDirectory and test the sign-on experience.

> **Tip**
>
> You can use groups to organize user identities, as explained in Groups in the PingOne documentation. You can also set access to applications, as explained in Application access control in the PingOne documentation.

Steps

- To set up SSO to the PingDirectory admin console from PingOne, follow the steps detailed in Setting up SSO to PingDirectory in the PingOne documentation.

  Result:

  You have succesfully done the following:

  - Set up a matching user between PingOne and the PingDirectory environments that allows the server's `All Admin Users` identity mapper to map the PingOne ID token to a PingDirectory server LDAP user.
  - Set up the OpenID Connect (OIDC) client.
  - Satisfied the configuration requirements for both the PingDirectory admin console and PingDirectory server.

    > **Note**
    >
    > The OIDC client and the PingDirectory configuration properties are both outlined in more detail in Single sign-on with the admin console.

Setting up SSO to PingDirectory from PingFederate

You can set up single sign-on authentication to the PingDirectory admin console using PingFederate as the authorization server.

To increase your control over who can access the PingDirectory admin console, you can configure single sign-on (SSO) using PingFederate as the OpenID Connect (OIDC) provider. Migrating authorization to PingFederate enables you to manage administrative access to PingDirectory more effectively in environments where you already rely on PingFederate workflows.

To set up SSO authentication to the admin console:

1. Complete the steps for Configuring PingFederate for SSO.
2. Complete the steps for Configuring PingDirectory for SSO.

Configuring PingFederate for SSO

Before you begin

- You must have a working PingFederate instance to serve as the OpenID Connect (OIDC) provider.

- To complete these steps, you must have an HTML form adapter instance. You can find details in Configuring an HTML Form Adapter instance in the PingFederate documentation.

About this task

To create an OIDC client and configure PingFederate as an authorization server, follow these steps.

Steps

1. Configure an access token manager.
    1. Go to Applications > OAuth > Access Token Management.
    2. Click Create New Instance.
    3. Enter or select the access token management instance properties as shown in the following table.

> **Important**
>
> Use the **Extend the contract** field to add the `admin_role` and `sub` attributes. Learn more about Defining the access token attribute contract in the PingFederate documentation.
> Configure other values as needed or leave the default values in place.

Tab	Property	Value
Type	Instance Name	`jwt`
	Instance ID	`jwt`
	Type	JSON Web Tokens
Instance Configuration	Certificates	Add your signing certificate and give it a key ID.
	Token Lifetime	`120`
	Use Centralized Signing Key	False
	JWS Algorithm	RSA using SHA-256
	Active Signing Certificate Key ID	Enter the key ID you assigned to your signing certificate.
	Enable Token Revocation	False
	Include Key ID Header Parameter	True
	Include X.509 Thumbprint Header Parameter	False
	Default JWKS URL Cache Duration	`720`

Tab	Property	Value
	Include JWE Key ID Header Parameter	True
	Include JWE X.509 Thumbprint Header Parameter	False
	Client ID Claim Name	`client_id`
	Scope Claim Name	`scope`
	Space Delimit Scope Values	True
	JWT ID Claim Length	`22`
	JWKS Endpoint Path	`/oauth/jwks`
	JWKS Endpoint Cache Duration	`720`
	Publish Key ID X.509 URL	False
	Publish Thumbprint X.509 URL	False
	Expand Scope Groups	False
Session Validation	Include Session Identifier In Access Token	True
	Check Session Validation Status	False
	Check Session Revocation Status	False
	Update Authentication Session Activity	False
Access Token Attribute Contract	Attribute	`admin_role`
	Attribute	`sub`
	Default Subject Attribute	USER_KEY
Access Control	Restrict Allowed Clients	False

4. Click Save.

2. Create an OIDC policy.

    1. Go to Applications > OAuth > OpenID Connect Policy Management.

    2. Click Add Policy.

3. Enter or select the OIDC policy properties as shown in the following table.

> **Note**
>
> Configure other values as needed or leave the default values in place.

Tab	Property	Value
Manage Policy	Policy ID	`jwtOIDCpolicy`
	Policy Name	`jwtOIDCpolicy`
	Access Token Manager	Select the jwt access token manager you previously created.
	ID Token Lifetime	`5`
	Include Session Identifier in ID Token	True
	Include User Info in ID Token	True
	Include State Hash in ID Token	False
	Return ID Token on Refresh Grant	False
	Reissue ID Token during Hybrid Flow	False
Attribute Contract	Attribute	`sub`
	Attribute	`admin_role`
Attribute Scopes	openid	`admin_role`
Contract Fulfillment	admin_role	Select Access Token in the Source menu and admin_role in the Value menu.
	sub	Select Access Token in the Source menu and sub in the Value menu.

4. Click Save.

3. Create a policy contract grant mapping.

    1. Go to Authentication > OAuth > Policy Contract Grant Mapping.

    2. In the Policy Contract menu, select your authentication policy contract and click Add Mapping.

    3. On the Attribute Sources & User Lookup tab, click Next.

4. On the Contract Fulfillment tab, for both the User_Key and User_Name, select Authentication Policy Contract for the Source and subject for the Value.
5. Complete any other configuration, as needed, and save the mapping.

4. Create an access token mapping between your authentication policy contract and the access token manager you previously created.

> **Important**
>
> If needed, create an authentication policy contract. The `sub` attribute is required.

1. Go to Applications > OAuth > Access Token Mappings.
2. On the Access Token Mappings page, in the Context menu, select your authentication policy contract.
3. In the Access Token Manager menu, select the jwt access token manager you previously created.
4. Click Add Mapping.
5. On the Attribute Sources & User Lookup tab, click Next.
6. On the Contract Fulfillment tab, do the following:
   - For the `admin_role` attribute, select Text for the Source and enter `fullAdmin` for the Value.
   - For the `sub` attribute, select Authentication Policy Contract for the Source and subject for the Value.
7. Complete any other configuration, as needed, and save the mapping.

5. Create an OIDC client.

1. Go to Applications > OAuth > Clients and click Add Client.
2. Enter or select the client application properties as shown in the following table.

> **Note**
>
> Configure other values as needed or leave the default values in place.

Property	Value
Client ID	`PingDirectoryConsole`
Client Name	`PingDirectoryConsole`
Description	`Client for the PingDirectory admin console`
Client Authentication	Select Client Secret.
Client Secret	Select the Change Secret check box and enter a password, or click Generate Secret and note the generated secret value.

Property	Value
Redirect URLs	Enter `https://<hostname>:<port>/console/oidc/cb`, supplying the hostname and port values for your PingDirectory server instance. **Note** To obtain the PingDirectory admin console port value, run `bin/status`.
Bypass Authorization Approval	Select Bypass.
Allowed Grant Types	Select Authorization Code.
Default Access Token Manager	Select jwt.
OpenID Connect > ID Token Signing Algorithm	Select RSA using SHA-256.
OpenID Connect > Policy	Select the OIDC policy you previously created.

3. Click Save.

6. Create an OAuth set authentication selector.

    1. Go to Authentication > Policies > Selectors.
    2. On the Selectors page, click Create New Instance.
    3. Enter or select the authentication selector properties as shown in the following table.

Tab	Property	Value
Type	Instance Name	`PD Console Selector`
	Instance ID	`PDConsoleSelector`
	Type	OAuth Client Set Authentication Selector
Authentication Selector	Clients	Select the OIDC client you previously created.

    4. Complete any other configuration, as needed, and save the selector.

7. Create and save an authentication policy by following steps 1-11 of Creating an authentication policy.

> **Important**
>
> In step 9, for the **Yes** option on the selector, select your HTML form adapter instance from the **IdP Adapters** menu.

Result

You have completed the PingFederate SSO configuration.

Configuring PingDirectory for SSO

Before you begin

You must have a working PingDirectory server instance that accepts LDAPS connections. This server will host the admin console being configured for SSO.

About this task

To enable administrators to use PingFederate to sign on to the PingDirectory admin console using SSO and configure PingDirectory to accept access and ID tokens from PingFederate, follow these steps.

Steps

1. **Create administrator accounts in PingDirectory.**
    1. **Make a list of the accounts that should have administrative access.**
    2. **For each account in the previous list, run the following command:**

    ```
    $ bin/dsconfig create-root-dn-user \
      --user-name <Username>  \
      --set first-name:<Given Name>  \
      --set last-name:<Family Name>
    ```

    Creating these administrators allows the `All Admin Users` identity mapper to map the PingFederate ID token to a PingDirectory LDAP user.
2. **Create an HTTP external server by running the following command:**

    ```
    $ bin/dsconfig create-external-server \
      --server-name PingFederateHttpServer \
      --type http \
      --set base-url:https://<PingFederate_server>:9031 \
      --set hostname-verification-method:allow-all
    ```

    Creating the HTTP external server enables an HTTP connection to PingFederate.
3. **Create an OIDC ID token validator by running the following command:**

```
$ bin/dsconfig create-id-token-validator \
  --validator-name PingFedTokenValidator \
  --type openid-connect \
  --set enabled:true \
  --set "identity-mapper:All Admin Users" \
  --set issuer-url:https://<PingFederate_server>:9031 \
  --set evaluation-order-index:1 \
  --set allowed-signing-algorithm:RS256 \
  --set openid-connect-provider:PingFederateHttpServer \
  --set jwks-endpoint-path:https://<PingFederate_server>:9031/pf/JWKS
```

Note

Set the `openid-connect-provider` value to the name of the HTTP external server you previously created.

Creating the ID token validator enables PingDirectory to validate the OIDC ID token it receives from PingFederate.

4. **Create a SASL mechanism handler by running the following command:**

```
$ bin/dsconfig create-sasl-mechanism-handler \
  --handler-name PingFedSASLHandler \
  --type oauth-bearer \
  --set enabled:true \
  --set id-token-validator:PingFedTokenValidator
```

Note

Set the `id-token-validator` value to the name of the OIDC ID token validator you previously created.

Creating the SASL mechanism handler enables PingDirectory to accept the access token it receives from PingFederate.

5. **Configure the Console web application extension.**

    1. **Run `bin/dsconfig`.**
    2. **Enter the number for the Web Application Extension configuration.**

    Note

    This option is only shown for the **Advanced** objects configuration menu. If needed, enter option **o** and change the configuration menu level.

    3. **To show existing web application extensions, enter `3`.**
    4. **To edit the Console web application extension, press enter.**
    5. **Configure the Console's properties as shown in the following table.**

    Note

    Configure other values as needed or leave the default values in place.

Property	Value
sso-enabled	true
oidc-client-id	Enter the client ID you provided for the PingFederate client application you previously created.
oidc-client-secret	Enter the client secret you provided for the PingFederate client application you previously created.
oidc-issuer-url	Enter the PingFederate OIDC token issuer URL. **Note** This value should match the value you provided for the `issuer-url` argument when creating the OIDC token validator.
ldap-server	This value should already be set, but you can confirm it by running `bin/status` on the PingDirectory server hosting the admin console for SSO connections.

6. Enter `f` after you complete your configurations.

Configuring the Console web application extension enables the PingDirectory admin console to accept ID token credentials for SSO from an OIDC ID token issued by your trusted PingFederate authorization server.

6. Restart PingDirectory.

7. Go to the PingDirectory admin console URL.

> **Note**
> The URL takes the form `https://<PingDirectory_server>:<HTTP_port>/console/` . If needed, you can run `bin/status` to find this information.

Result:

You should be redirected to the PingFederate sign-on form.

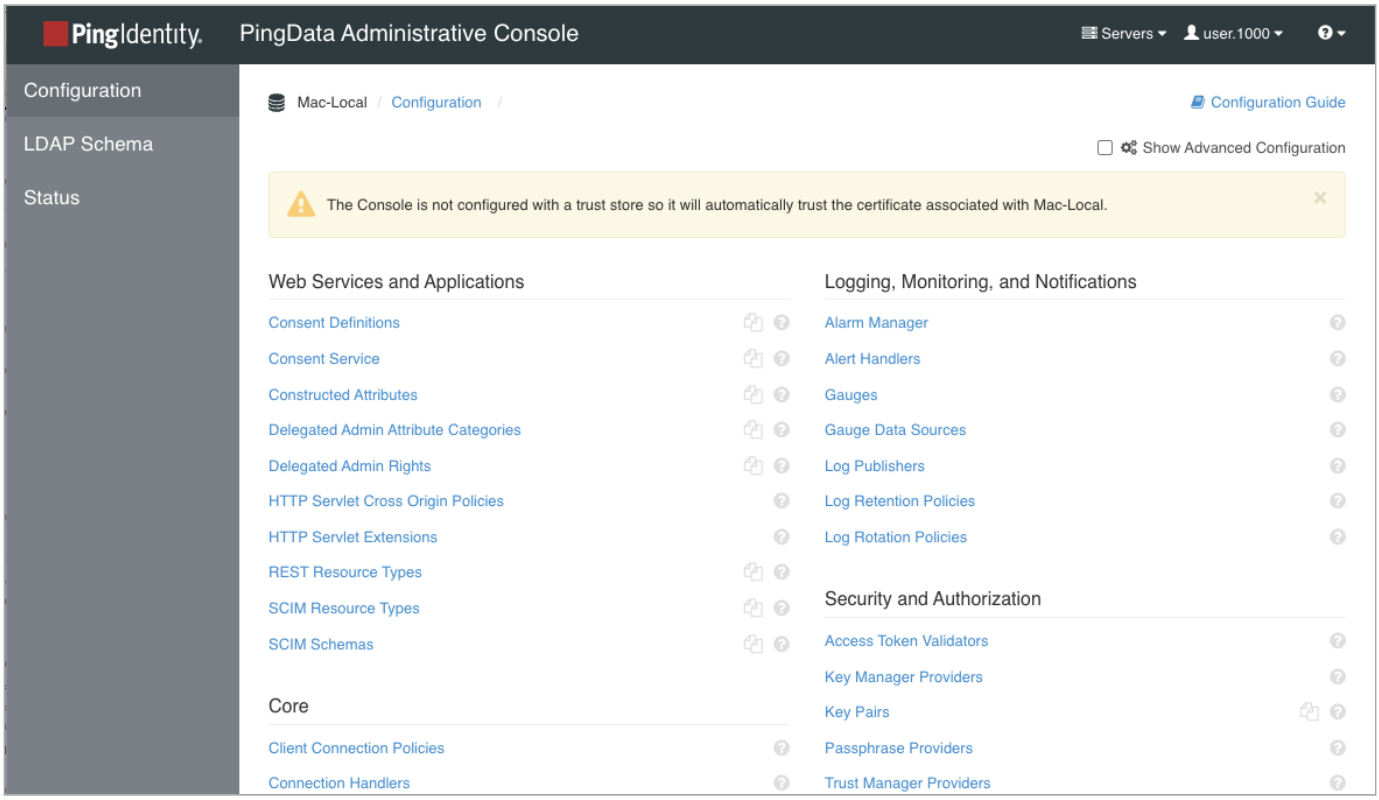
Troubleshooting:

If you have previous admin console sessions, close your browser tabs and retry or use your browser's privacy mode.

8. Enter the credentials for one of the administrator accounts you created in step 1.

Result:

You should be signed on to the PingDirectory admin console.



Setting up SSO to PingDirectory from a generic OpenID Connect provider

To set up single sign-on (SSO) access to the PingDirectory admin console from your OpenID Connect (OIDC) provider, configure the OIDC provider and PingDirectory and test the sign-on experience.

Before you begin

Ensure that you have:

- A PingDirectory server that accepts LDAPS connections

  This server will host the admin console that is being configured for SSO.

- An OIDC provider that supports OIDC discovery

About this task

For more information on the configuration properties you are setting up with this task, see Single sign-on with the admin console.

Steps

1. Configure your OIDC provider to access the PingDirectory admin console:
   1. Set the redirect URL to `https://<hostname>:<port>/console/oidc/cb` where `<hostname>` and `<port>` are for the PingDirectory server.
   2. Set the JSON web token (JWT) signing algorithm.
   3. Record the client ID, client secret, and issuer URL for use in step 5.

2. Create a token validator on the PingDirectory server by running a command similar to the following.

   Example:

   ```
   dsconfig create-id-token-validator \
        --validator-name "OpenID Token Validator" \
        --type openid-connect \
        --set enabled:true \
        --set "identity-mapper:All Admin Users" \
        --set evaluation-order-index:1 \
        --set issuer-url:<OIDC_Provider_Issuer_URL>
        --set allowed-signing-algorithm:<JWT_signing_algorithm>
   ```

   Provide your own values for `<OIDC_Provider_Issuer_URL>` and `<JWT_signing_algorithm>`, where the algorithm is the one you set in the previous step.

   Although not shown in the example, the command must also set the properties in one of the rows in the following table.

Properties	Descriptions
`openid-connect-provider` and `jwks-endpoint-path`	An OpenID Connect provider, which refers to an HTTP External Server, and a JWKS (JSON web key set) endpoint path
`signing-certificate`	A signing certificate

3. To create an LDAP user in `cn=Root DNs,cn=config` that the OIDC provider can send an ID token for, use the following `dsconfig` command.

   If, in the previous step, you use the `All Admin Users` identity mapper and the ID token validator's `subject-claim-name` is `sub` (the default), then the `sub` value of the ID token that the OIDC provider sends must be the `cn` of an admin user on the PingDirectory server. For example, assume the OIDC provider sends an ID token with the claim `sub=admin-user`. Then, there must be an LDAP user in `cn=Root DNs,cn=config` or in `cn=Topology Admin Users,cn=topology,cn=config` who has `cn=admin-user`.

   ```
   dsconfig create-root-dn-user --user-name admin-user
   ```

4. Create a SASL mechanism handler on the PingDirectory server to use the validator you just created by running a command similar to the following.

   Example:

   ```
   dsconfig create-sasl-mechanism-handler \
        --handler-name OAUTHBEARER \
        --type oauth-bearer \
        --set enabled:true \
        --set "id-token-validator:OpenID Token Validator" \
        --set require-both-access-token-and-id-token:false
   ```

5. Run the following command, substituting values for the ID, secret, and issuer URL.

```
dsconfig set-web-application-extension-prop \
     --extension-name Console \
     --set sso-enabled:true \
     --set oidc-client-id:<OIDC_Client_ID>  \
     --set oidc-client-secret:<OIDC_Client_Secret>  \
     --set oidc-issuer-url:<OIDC_Provider_Issuer_URL>
```

6. To finalize your changes, disable and re-enable the HTTPS Connection Handlers with the following commands.

```
dsconfig set-connection-handler-prop \
     --handler-name "HTTPS Connection Handler" \
     --set enabled:false

dsconfig set-connection-handler-prop \
     --handler-name "HTTPS Connection Handler" \
     --set enabled:true
```

7. Test the sign on.

   Result:

   The admin console should open.

Working with the schema

A schema is the set of PingDirectory 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 PingDirectory server objects.

The PingDirectory server includes a default set of read-only schema files that define the core properties for the server. The admin console provides a schema editor to view existing schema definitions and add new custom schema elements to your DIT.

> **Note**
>
> Any attempt to alter a schema element defined in a read-only file or add a new schema element to a read-only file results in an `Unwilling to Perform` result.

About the schema editor

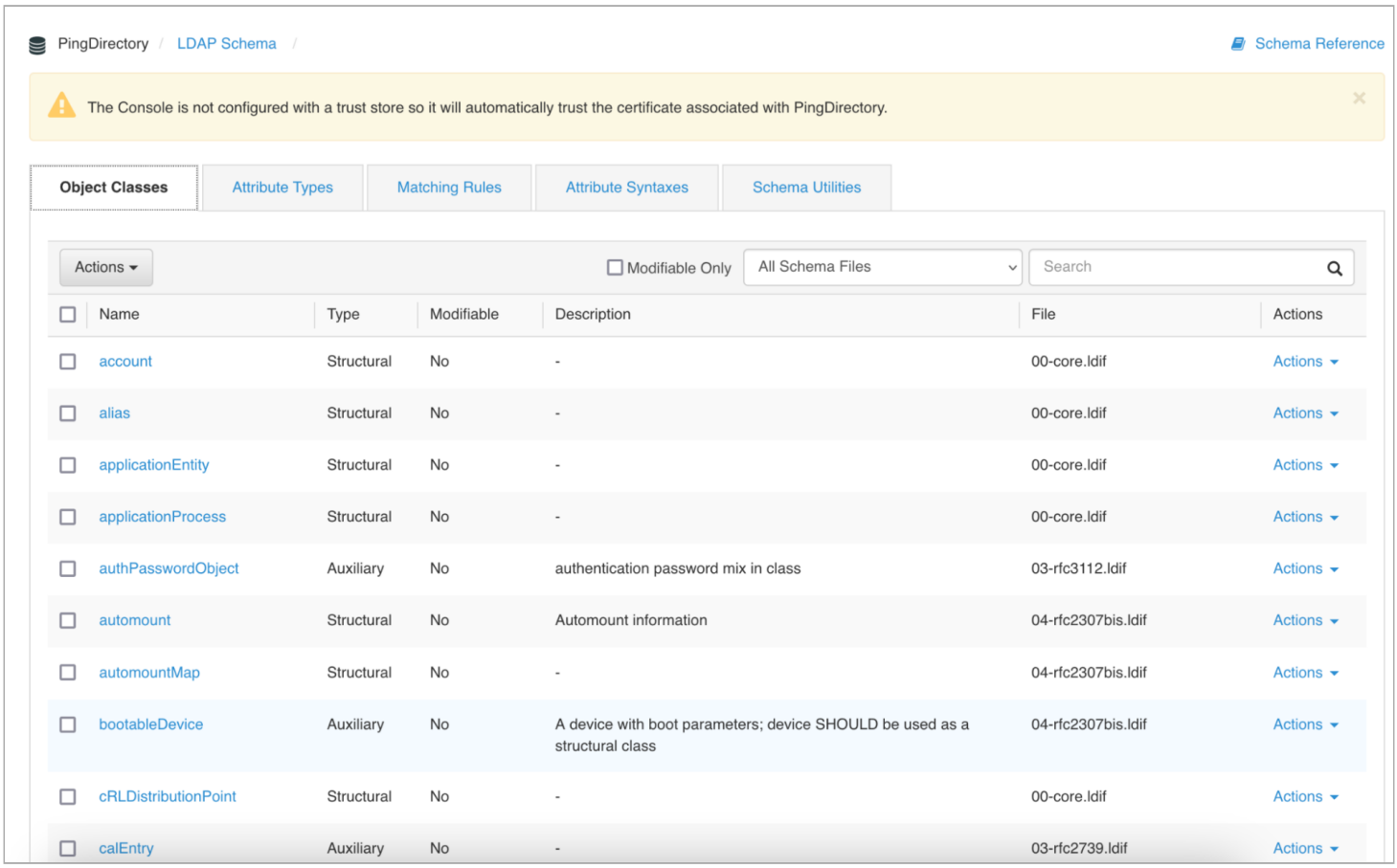
The admin console provides a user-friendly graphical editor with tabs to manage any existing schema components related to the directory information tree (DIT).

The list of existing schema components that can be managed includes:

- Object classes
- Attributes
- Matching rules
- Attribute syntaxes
- Schema utilities

The following list describes the tabs of the Schema Editor page:

- 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 supports schema file imports by first checking for proper syntax compliance and generating any error message if the definitions do not meet specification.



Example schema editor page

The schema editor provides two views for each definition:

- The Properties View that breaks down the schema definition by its properties and shows any inheritance relationships among the attributes.
- The LDIF View that shows the equivalent schema definition in ASN.1 format, which includes the proper text spacing required for each schema element.

Using the schema editor utilities

The schema editor provides a Schema Utilities tab where you can import new schema elements from a file and check schema compliance.

About this task

If you are importing a schema file, the system automatically checks for compliance before the import. If the definition doesn't meet schema compliance, the system displays an error message. However, you should check if your file is compliant with your schema before importing it.

To check schema compliance using the schema editor:

Steps

1. Start the admin console.
2. In the top-level navigation menu, click LDAP Schema.
3. In the schema editor, click the Schema Utilities tab.
4. Add your schema definition using one of two methods:

   Choose from:

   - To have the system check an LDIF file, click Import Schema Elements and select a file to upload.
   - Copy and paste a new schema definition into the field.
5. Click Validate Entries.

   Result:

   If there is a problem with your definition, you see an error message.

Modifying the schema

You can create new attribute types or object classes in the schema editor.

When extending the schema, you should not modify native schema elements. Click the corresponding tab for the task that you want to perform.

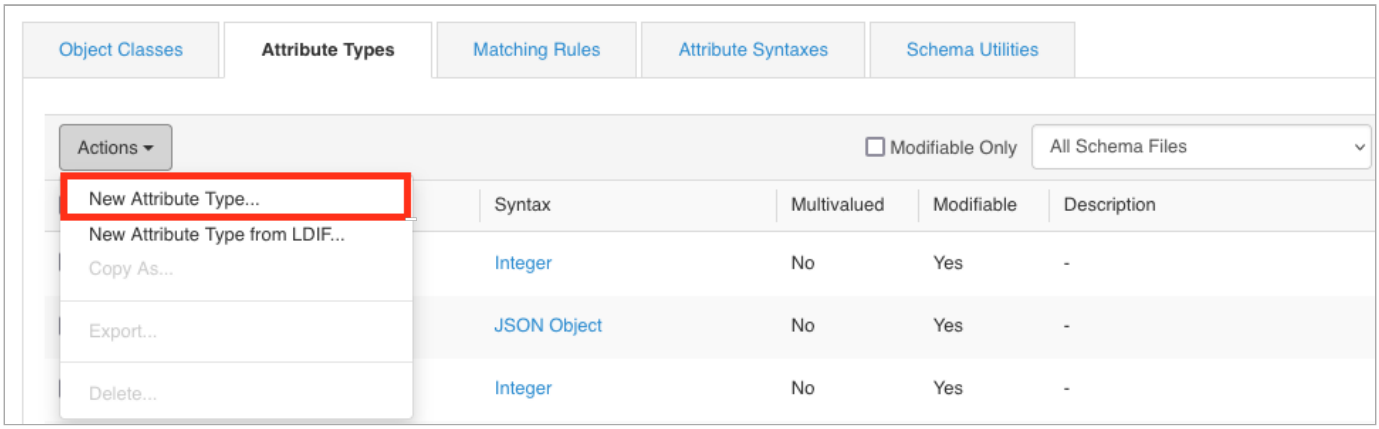
Creating a new attribute type using the schema editor

About this task

You can create attribute types for any new properties that logically belong to objects in your directory. To create a new attribute type:

Steps

1. Start the admin console.
2. In the top-level navigation menu, select LDAP Schema.
3. Click the Attribute Types tab, and then in the Actions list, select New Attribute Type.



4. Enter the properties for the new attribute types according to the help text.
5. Click Save.

Creating a new object class using the schema editor

Before you begin

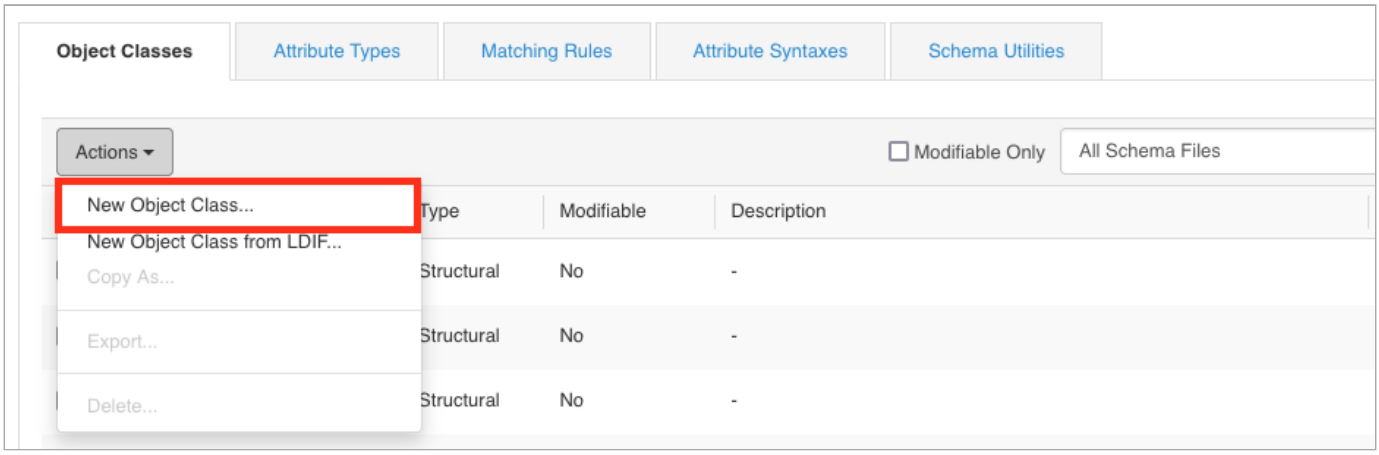
Make sure that any attributes that are part of the new object class are defined before creating the object class.

About this task

Creating a new object class is similar to creating a new attribute. To create a new object class:

Steps

1. Start the admin console.
2. In the top-level navigation menu, select LDAP Schema.
3. Click the Object Classes tab, and then in the Actions list, select New Object Class.



4. Enter the properties for the new object class.
5. 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.

> **Note**
>
> All custom attributes appear at the bottom of the list in the **Attributes** box.

Modifying a schema definition

About this task

You can only edit schema definitions in the PingDirectory server that are read-write. The 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 admin console.
2. In the top-level navigation menu, click Schema.
3. Click the Object Classes tab.
4. Select the object class that you want to modify, and then click Actions → Edit.

5. Make your changes and click OK.

Deleting a schema definition

About this task

You can only delete schema definitions in the PingDirectory server that are read-write. In general, you can only remove schema definitions in the Custom folder of the schema editor. Ensure that the schema element you are deleting is not currently in use.

To delete a schema definition:

Steps

1. Start the admin console.
2. In the top-level navigation menu, click Schema.
3. Click the Object Classes tab.
4. Select the object class that you want to remove, and then in the Actions list, select Delete.

Result:

The Confirmation dialog box opens.

5. To delete the schema element, click Yes.

Managing tasks

The PingDirectory server has a tasks subsystem that allows you to schedule basic administrative operations.

Although the server provides a full suite of command line tools, several of these operations can be executed directly from the admin console. Learn more in the following:

- Collecting support data from the admin console
- Generating the server profile from the admin console
- Creating a file based access log publisher
- About recurring tasks and task chains
- Scheduling LDIF export as a recurring task

Collecting support data from the admin console

You can collect support data to aggregate data and send it to a support provider.

About this task

PingDirectory servers provide information about their current state and any problems encountered. If a problem occurs, run the Collect Support Data tool. 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.

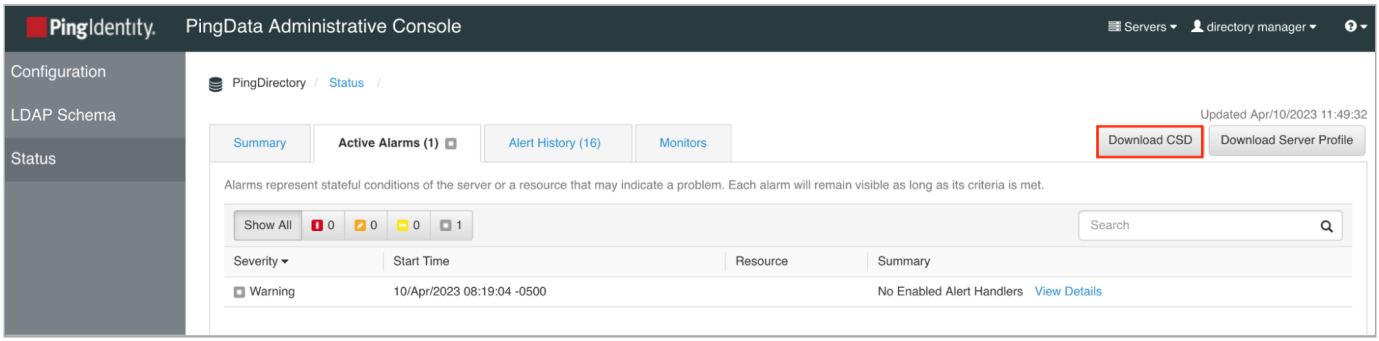
The data collected by the Collect Support Data tool can vary between systems. The data collected includes the configuration directory, summaries and snippets from the logs directory, an LDIF of the monitor and RootDSE entries, and a list of all files in the server root.

> **Important**
>
> The Collect Support Data task is not recommended on directory servers running in a Docker container because this container has limited memory. If running the Collect Support Data task exhausts the container's memory, then the server will crash.

To run this tool from the admin console:

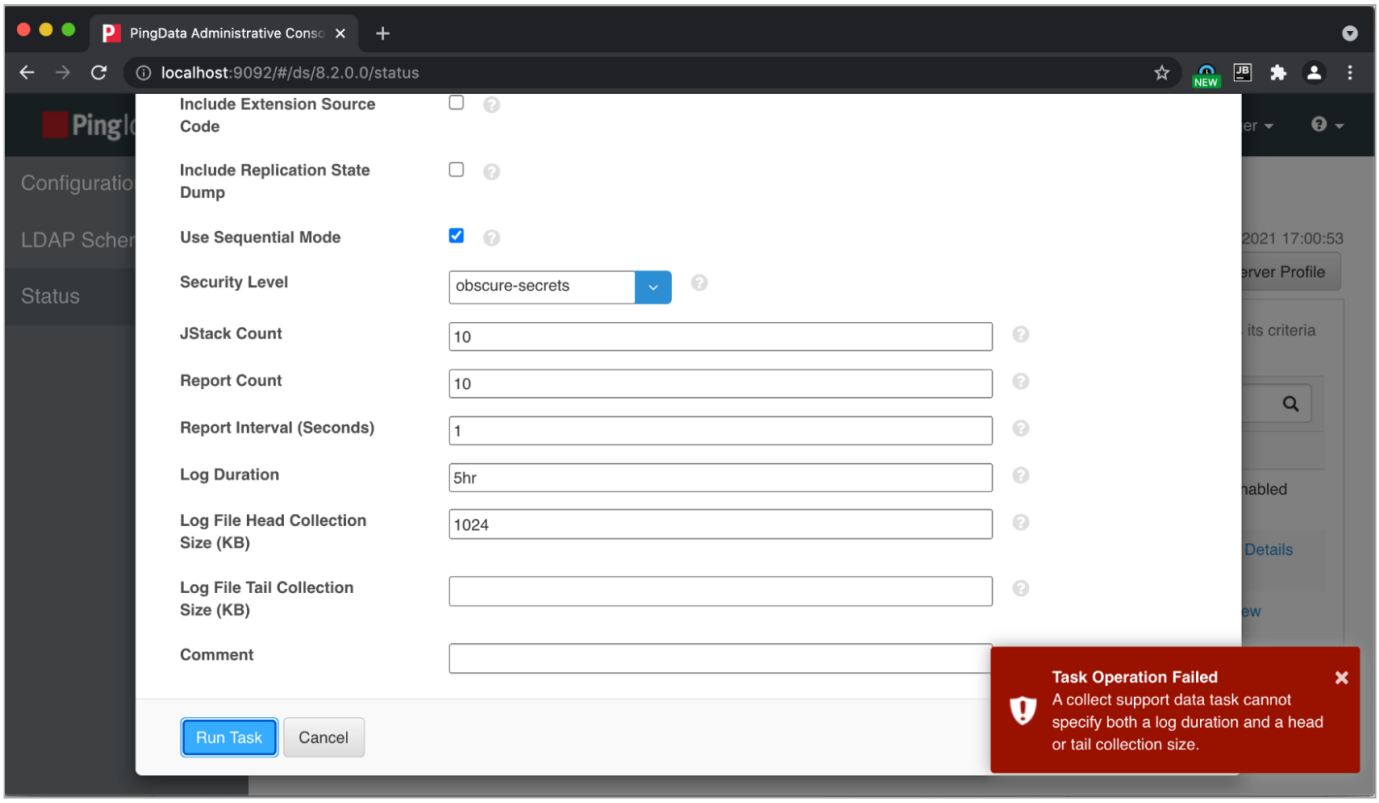
Steps

1. **Start the admin console.**
2. **In the top-level navigation menu, select Status.**
3. **Click Download CSD.**



4. **In the Collect Support Data window, populate the properties according to the help text.**
5. **Click the Run Task button.**

> **Note**
>
> If there is any issue with scheduling the task, an error message displays in the bottom-right corner of the page. Resolve the issue as described to proceed.



6. If the task is scheduled successfully, a confirmation message appears in the bottom-right corner of the page.
7. Wait for the task to complete.

   This can take several minutes.
8. After the task is complete, the generated files are automatically downloaded to the `<directory>/csd-files` directory.

> **Note**
>
> When the admin console sends a request to the PingDirectory server to generate the support files, the `<directory>/csd-files` is the default path for storing the resultant file package. This default path can be modified in the `<directory>/console/WEB-INF/classes/application.yml` directory.

Generating the server profile from the admin console

Run the Download Server Profile task to generate a format for setting up and configuring new server instances.

About this task

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

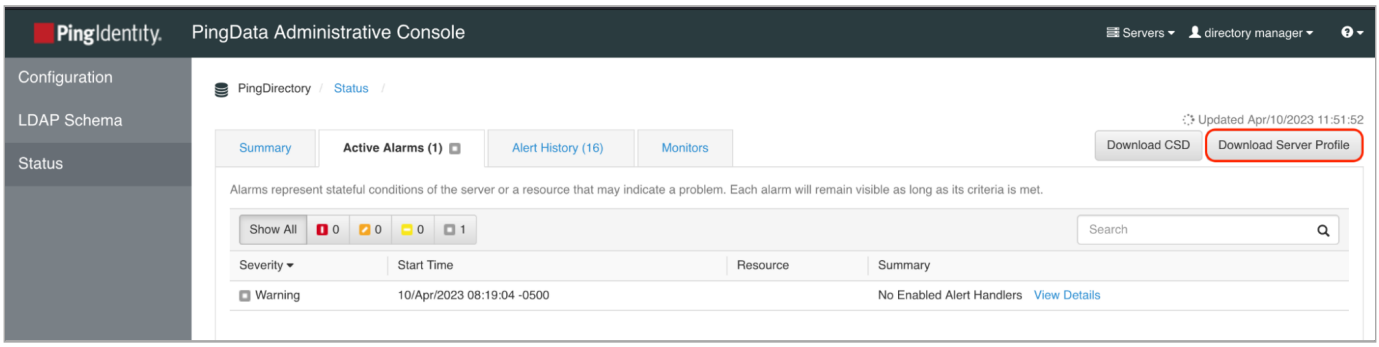
- Server SDK extensions
- Additional miscellaneous files

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. For more information, see Server profiles.

To run this tool from the admin console:

Steps

1. Start the admin console.
2. In the top-level navigation menu, select Status.
3. Click Download Server Profile.



4. In the Generate Server Profile window, enter path values according to the help text.
5. Click the Run Task button.
6. Wait for the task to complete.

   This can take several minutes.
7. After the task is completed, the server profile is downloaded to the `<directory>/profile-files` directory.

> **Note**
>
> When the admin console sends a request to the PingDirectory server to generate the server profile, `<directory>/profile-files` is the default path for storing the resultant file package. This default path can be modified in the `<directory>/console/WEB-INF/classes/application.yml` directory.

Creating a file based access log publisher

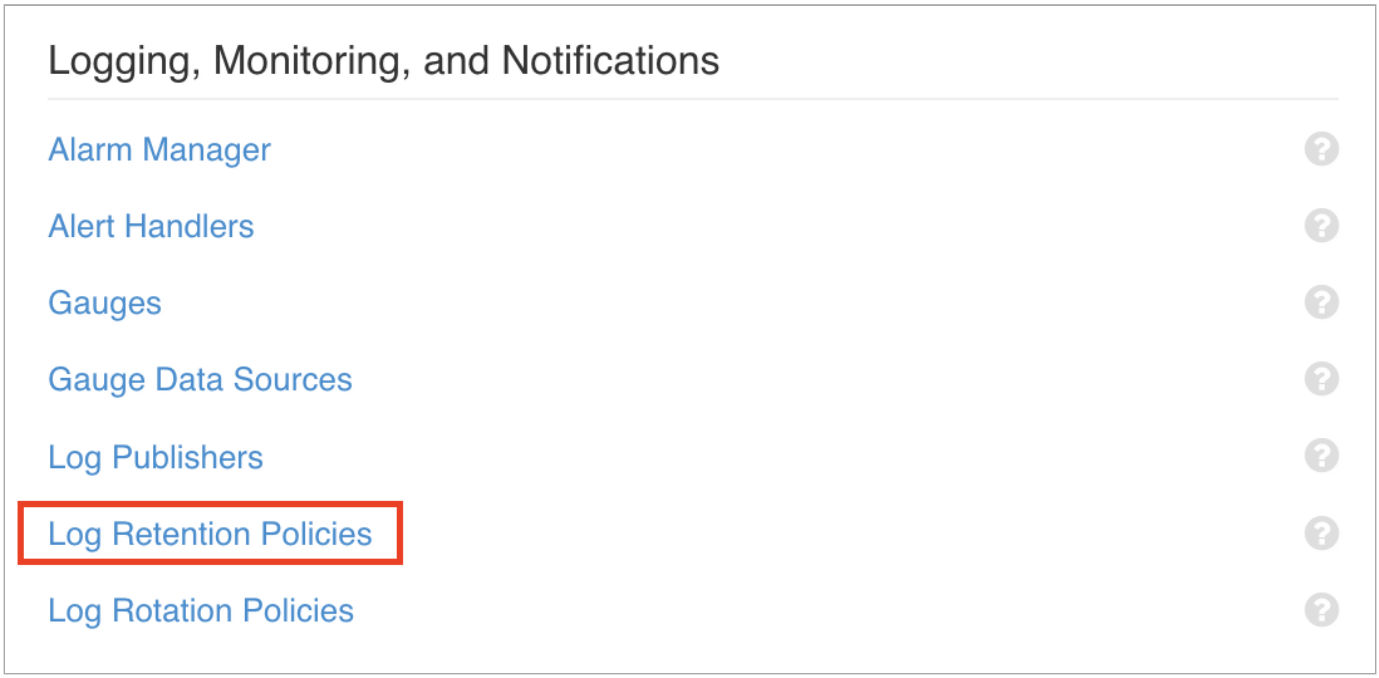
The PingDirectory server supports different types of log publishers that can be used to provide monitoring information for operations, access, debug, and error messages that occur during normal server processing.

About this task

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. The filed based access log publisher publishes access messages with information about the operations processed by the server. For more information, see Types of log publishers.

Steps

1. Start the admin console.
2. In the main menu, select Configuration, and in the Logging, Monitoring, and Notifications section, select Log Retention Policies.



3. In the New Log Retention Policy list, select your desired policy and fill out the fields according to the help text.
4. Click Save.
5. Return to the Logging, Monitoring, and Notifications section and select Log Rotation Policies.
6. In the New Log Rotation Policy list, select your desired policy and fill out the fields according to the help text.
7. Click Save.
8. Return to the Logging, Monitoring, and Notifications section and select Log Publishers.
9. In the New Log Publisher list, select File Based Access Log Publisher.
10. Fill out the fields according to the help text. In the Rotation Policy field of the Log File Management section, select your desired rotation policy in the Available list, and click the Add Value button to activate the policy.
11. Repeat step 10 for the Retention Policy field.
12. Click Save.

About recurring tasks and task chains

You can use the admin console to create recurring tasks and task chains to perform regular maintenance tasks for the 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 recurring tasks and 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.

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.
> You cannot 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.

Scheduling LDIF export as a recurring task

You can create an LDIF export recurring task and add it to a recurring task chain to automate the export with a custom schedule.

About this task

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.

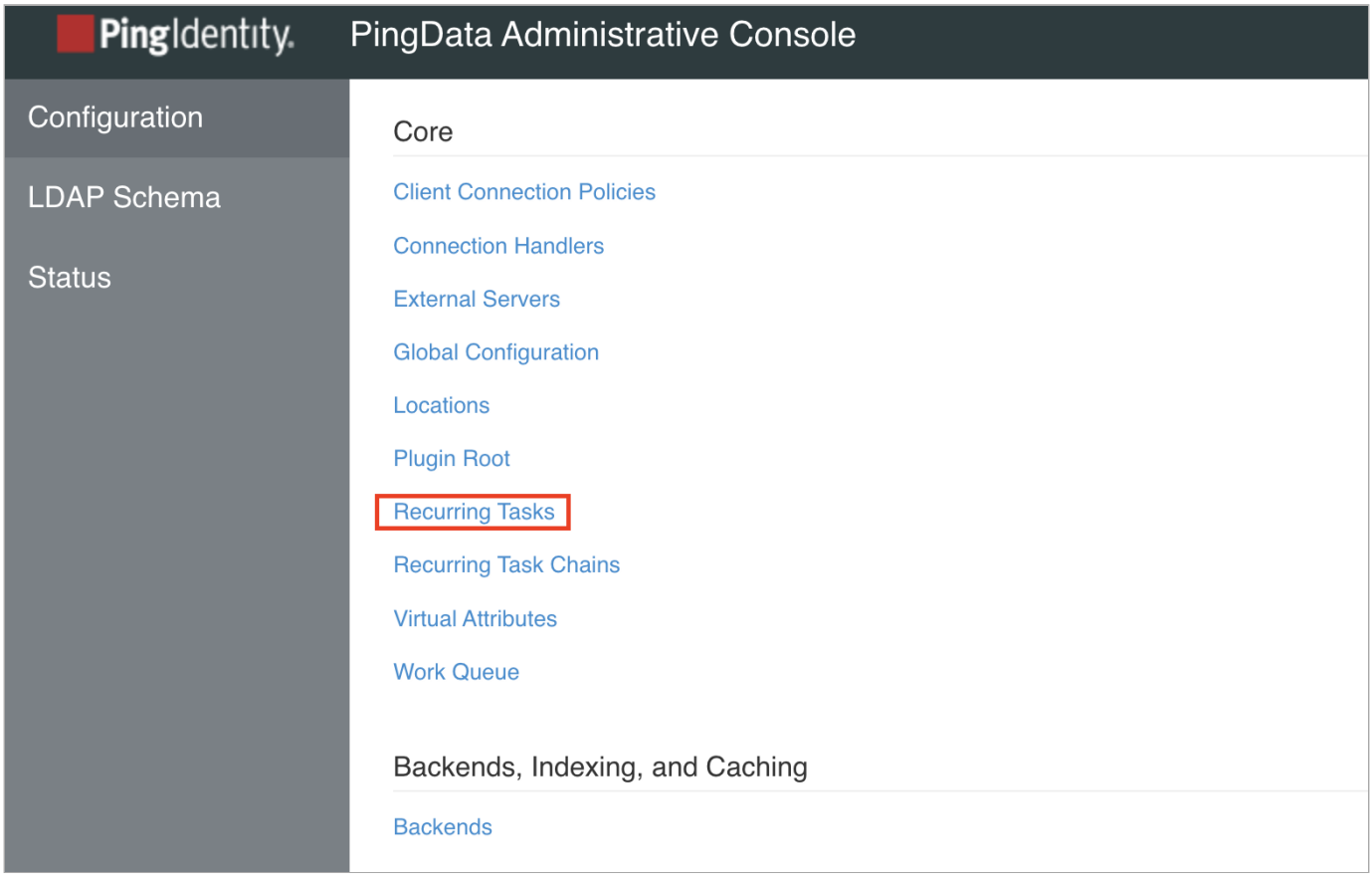
> **Note**
>
> Public backends are non-administrative backends containing user-supplied data.

If the global configuration is set to encrypt LDIF exports by default, the server compresses and encrypts the LDIF exports. If encryption is configured during server setup, LDIF exports are encrypted by default. 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.

To create a recurring LDIF export task:

Steps

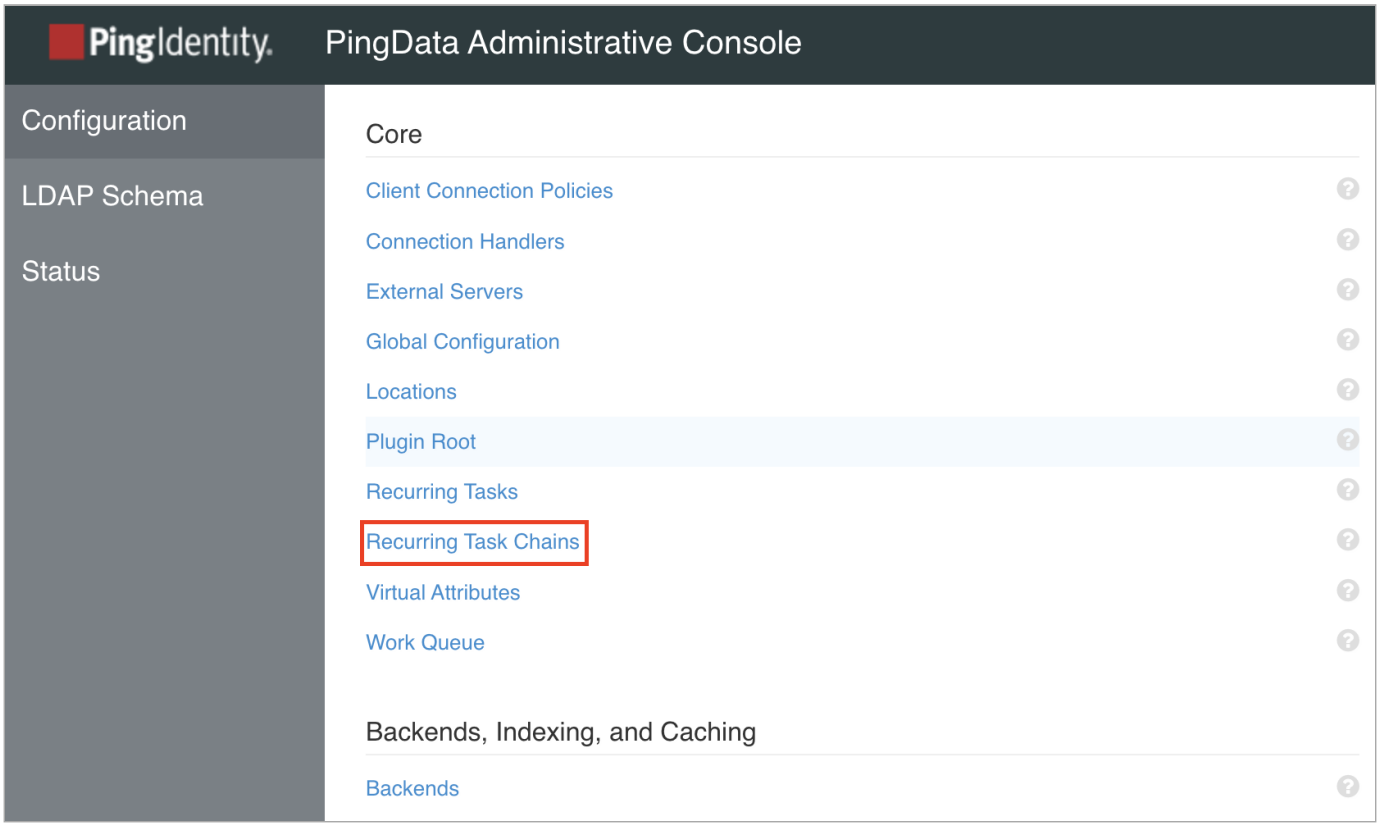
1. Start the admin console.
2. In the top-level navigation menu, select Configuration, and in the Core section, select Recurring Tasks.



3. In the New Recurring Task list, select LDIF Export Recurring Task.
4. In the Name field, enter the desired name of the task.
5. Populate the rest of the task's properties according to the help text.

> **Note**
>
> You must configure a value for either the **Retain Previous LDIF Export Count** field, which specifies the minimum number of previous LDIF exports preserved after a new export, or the **Retain Previous LDIF Export Age** field, which specifies the minimum age of previous LDIF exports preserved after a new export. If each of these properties has a value, then only LDIF exports failing to satisfy both criteria are candidates for removal.

6. Click Save.
7. In the Core section, in the Configuration menu, click Recurring Task Chains.



8. Click New Recurring Task Chain.
9. In the Name field, enter the desired name of the task chain.
10. In the Recurring Task section, select your newly created recurring task and any other tasks you want to schedule.
11. Enter the scheduling details according to the help text.
12. Click Save.

Monitoring with the admin console

The admin console can be used to monitor items, such as disk space usage, active operations in the server, and alarms raised.

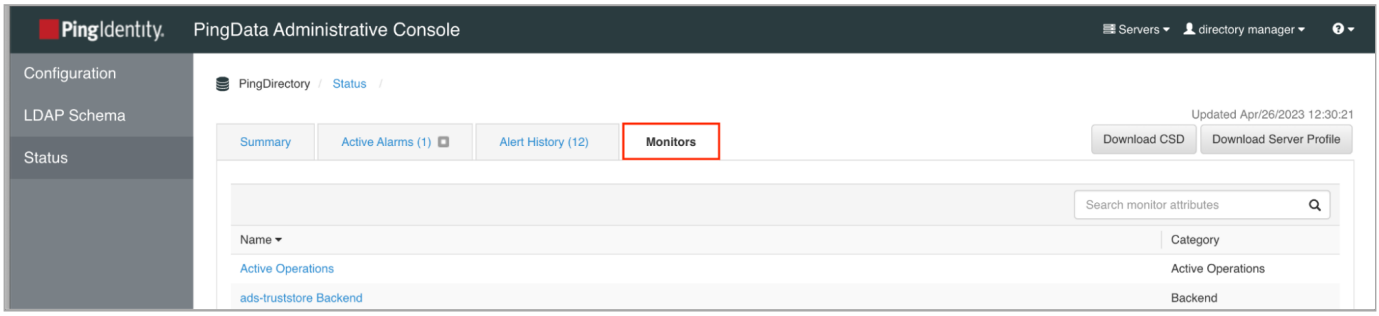
About this task

The console provides a status option that accesses the server's monitor content.

To view the monitor dashboard:

Steps

1. In your browser, go to http://<*server-name*>:<*port*>/console.
2. Enter the root user distinguished name and password. Click Login.
3. In the top-level navigation menu, select Status.
4. Click the Monitors tab.



Result

The Monitors page opens, and you can monitor your server by selecting a monitor attribute.

Removing the admin console

You can remove the admin console from your PingDirectory deployment.

About this task

You can configure and operate the PingDirectory server without the admin console by using the command-line interface (CLI). There aren't any server functions available from the admin console that aren't also available through the CLI.

Steps

Follow these steps to remove the admin console from your PingDirectory deployment:

1. Stop the server.

   Example:

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

2. Remove the `console.war` file from the `webapps` directory.
3. Edit the `config/config.ldif` file as follows:
   1. In the `dn: cn=HTTPS Connection Handler,cn=Connection Handlers,cn=config` section, remove the following line:

```
ds-cfg-web-application-extension: cn=Console,cn=Web Applications,cn=config
```

2. In the `dn: cn=Console,cn=Web Applications,cn=config` section, remove the following lines:

```
objectClass: ds-cfg-console-web-application-extension
objectClass: ds-cfg-web-application-extension
objectClass: top
cn: Console
ds-cfg-description: The Administrative Console
ds-cfg-log-file: logs/webapps/console.log
```

3. Save your changes.

4. Restart the server.

Example:

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

Result:

You have successfully removed the admin console from your PingDirectory deployment.

> **Note**
>
> Upgrading the server after removing the admin console restores the `console.war` file but doesn't affect the console configurations that you removed from `config.ldif`. The admin console won't run with these configurations removed. Optionally, you can remove `console.war` after completing a server upgrade.

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 PingDirectory 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 PingDirectory 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 PingDirectory 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 PingDirectory 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 PingDirectory 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 PingDirectory 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 PingDirectory 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 PingDirectoryProxy server. The soft-deleted entry is routed directly to the underlying PingDirectory server. There is one exception: in an entry-balancing deployment, the PingDirectoryProxy server is responsible for routing the soft-deleted entry to the PingDirectory 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 PingDirectory 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 Configuring a user to use soft or hard delete controls. 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

Steps

- Use any of the following methods to configure soft deletes on the PingDirectory 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. For more information, see Configuring soft deletes as a global configuration.
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. For more information, see Configuring soft deletes by connection criteria.
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. For more information, see Configuring soft deletes by request criteria. **Note** You can define both connection criteria and request criteria. When both criteria exist within a soft-delete policy, connection criteria is processed first, then 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.

Example:

```
$ 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.

   Example:

```
$ 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. Add the following ACIs to the base suffix or other point in the directory information tree (DIT) to restrict the scope as required.

   Example:

```
(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`.

   Example:

```
$ ./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.

  Example:

```
$ bin/ldapsearch \
  --baseDN entryUUID=4e9b7847-edcb-3791-b11b-7505f4a55af4+uid=user.1,ou=People,dc=example,dc=com \
  --searchScope base "(objectClass=*)"
```

  Result:

```
# 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.

  Example:

```
$ 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`.

   Example:

```
$ 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`.

   Example:

```
$ 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.

   Example:

   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.

  Example:

  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
```

Result:

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 PingDirectory 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.

   Example:

   ```
   $ 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`.

   Example:

```
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
```

Result:

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.

> **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.

  Example:

```
$ 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 PingDirectory server.

Steps

- To permanently remove a soft-deleted entry, run `ldapdelete` on the soft-deleted entry.

  Example:

  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 PingDirectory server.

Steps

- To permanently remove a soft-deleted entry run **`ldapdelete`** with the **`--useHardDelete`** subcommand on the soft-deleted entry.

  Example:

  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 PingDirectory 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`.

   Example:

   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.

   Example:

```
$ 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.

  Example:

```
$ 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 are treated as a soft delete.

The presence of a soft delete by connection criteria is exclusive of the soft delete by request criteria.

> **Note**
>
> Both a soft delete by connection criteria and a soft delete by request criteria can be present in a soft delete policy.

Enabling soft deletes by request criteria

Before you begin

Configure a soft-delete policy and global configuration.

Steps

1. To configure request criteria for soft deletes, use the `create-request-criteria` option with `dsconfig`.

Example:

In this example, 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)"
```

2. In the soft delete policy you previously created, set the `auto-soft-delete-connection-criteria` property to the simple criteria created in the previous step.

   Example:

   ```
   $ 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.

  Example:

  ```
  $ 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, PingDirectory server retains soft-deleted entries indefinitely.

For companies that want to set up automatic purging of soft-deleted entries, the server provides two configurable properties on the soft delete policy: the maximum retention time for all soft-deleted entries and the retained number of soft-deleted entries.

> **Note**
>
> These changes take effect without requiring a server restart.

Configuring soft-delete automatic purging

About this task

To enable automatic purging, you can change the retention time, the retained number of entries, or both. 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), can be preceded by an integer to specify a quantity for that unit, such as "1 d", "52 w", etc.

> **Note**
>
> After you configure the properties, the changes take effect immediately without the need for a server restart. The server deletes 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. To retrieve the name of the soft delete policy in effect, use the `get-global-configuration-prop` option with `dsconfig`.

   Example:

   In this example, the soft delete policy is called `default-soft-delete-policy`.

   ```
   $ bin/dsconfig get-global-configuration-prop \
     --property soft-delete-policy
   ```

2. To set the retention time or retained number of soft-deleted entries, use the `set-soft-delete-policy-prop` option with `dsconfig`.

   Example:

   This example shows how 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"
   ```

   Example:

   This example shows how 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. If it has not been assigned yet, assign the soft delete policy to the global configuration.

   Example:

   This example assigns `default-soft-delete-policy` to the global configuration.

```
$ bin/dsconfig set-global-configuration-prop \
  --set soft-delete-policy:default-soft-delete-policy
```

Disabling soft-delete automatic purging

About this task

The change takes effect immediately without the need of a server restart. If the server is in the middle of an automatic soft-delete purging, it might continue to purge entries until the next time it evaluates the soft delete policy.

Steps

- To disable soft-delete automatic purging, use the `set-soft-delete-policy` option with `dsconfig`.
- To reset the soft delete policy properties that control automatic purging, use the `set-soft-delete-policy-prop` with `dsconfig`.

  Example:

  This example resets the `soft-delete-retention-time` and `soft-delete-retain-number-of-entries` properties.

  ```
  $ 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.

Action	Result
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.
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 Object Identifier (OID) 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	Step 1.3.6.1.4.1.30221.2.5.20
Soft Delete Response Control	Step 1.3.6.1.4.1.30221.2.5.21
Hard Delete Request Control	Step 1.3.6.1.4.1.30221.2.5.22
Soft Undelete Request Control	Step 1.3.6.1.4.1.30221.2.5.23
Soft Delete Entry Access Control	Step 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 PingDirectory 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 PingDirectory server must be configured to allow soft deletes.

Operation	Options
`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 PingDirectory 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
Hard Delete Request Control	harddelete
Soft Undelete Request Control	undelete
Soft Delete Entry Access Control	softdeleteentryaccess

Monitoring soft deletes

The 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.

  Example:

  ```
  $ bin/ldapsearch --baseDN "cn=userRoot Backend,cn=monitor" \
    --searchScope sub "(objectclass=ds-backend-monitor-entry)"
  ```

  Result:

  ```
  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.

   Example:

```
$ 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.

   Example:

```
# 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=*)" "+"
```

   Result:

```
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 PingDirectory 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.

   Example:

   ```
   $ bin/dsconfig set-global-configuration-prop --reset soft-delete-policy
   ```

2. To reset the global configuration property, run the following `dsconfig` command.

   Example:

   ```
   $ 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 LDIF.

You can use the `bin/import-ldif` and `bin/export-ldif` tools to create or export database backends for online or 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 LDIF file. You can import all or a portion of the entries contained in the LDIF file, including a subset of the entries, a subset of the attributes within entries, or both. The command also supports importing data that has been compressed, encrypted or digitally signed, or both.

You can run `import-ldif` 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, is removed from service and data held in that backend is not available to clients.

> **Note**
>
> The `import-ldif` tool guards against the accidental overwriting of existing backend data. When importing data into a backend with a branch that already contains entries, to overwrite the entries in the branch, you must include the `--overwriteExistingEntries` option. If a branch contains just a single base entry, you don't need to include this option. The tool also rejects any entries that contain duplicate values within the same attribute. To make `import-ldif` treat these types of entries as if each attribute value is only provided once, include the `--ignoreDuplicateAttributeValues` option.

Validating an LDIF file

Before importing data, you can validate an import file using the PingDirectory server's `validate-ldif` tool.

About this task

The tool binds to the PingDirectory server locally or remotely and validates the LDIF file to determine whether it violates the server's schema. Elements that do not conform to the schema are rejected and written to standard output. You can specify the output filepath to which the rejected entries and reasons for their rejection are written. The `validate-ldif` tool works with regular non-compressed LDIF files or gzip-compressed LDIF files.

Steps

- To validate an LDIF file, run the `validate-ldif` tool.

  > **Note**
  >
  > Make sure the server is online before running this command.

  To process large files faster, you can 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.

  Example:

```
$ bin/validate-ldif --ldifFile /path/to/data.ldif \
  --rejectFile rejectedEntries
```

1. (Optional) To view the arguments, use the `--help` option.

   Result:

   ```
   1 of 200 entries (0 percent) were found to be invalid.
   1 undefined attributes were encountered.
   Undefined attribute departmentname was encountered 1 times.
   ```

About the database cache estimate

After successful completion of an import, the `import-ldif` command lists detailed information about the database cache usage characteristics of the imported data set.

To guide decisions for changing Java virtual machine (JVM) size and database-cache-percent for the backend, the current server configuration is considered along with the capabilities of the underlying hardware.

The `/logs/tools` directory contains additional files that describe the database cache characteristics in more detail.

Tracking skipped and rejected entries

During import, skip entries if they do not belong in the specified backend or if they are part of an excluded base distinguished name (DN) or filter.

Steps

- To write skipped entries to a specified file, use the `--skipFile {path}` option on the command line.

  You can add a comment indicating why the entries were skipped.

- To write information about rejected entries and the reasons for rejection to a specified file, use the `--rejectFile {path}` option.

  You can reject an entry if:

  - It violates the server's schema constraints.
  - Its parent entry does not exist.
  - Another entry already exists with the same DN.
  - It was rejected by a plugin.

Running an offline import

To import LDIF data encoded with the UTF-8 character set, run the `import-ldif` tool offline.

This data can come from LDIF files, compressed LDIF files (GZIP format), or from data generated with a MakeLDIF template.

> **Note**
>
> You do not need to authenticate as an administrator when performing offline LDIF imports.

Performing an offline import

Steps

1. Confirm the PingDirectory server is offline.
2. To import data from an LDIF file, use the `import-ldif` command.

   Do not specify any connection arguments when running the command.

   Example:

```
$ 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

1. Confirm the PingDirectory server is offline.
2. To import data from a compressed gzip formatted file, use the `import-ldif` command.

   > **Note**
   >
   > You must include the `--isCompressed` option to indicate that the input file is compressed.
   > Do not specify any connection arguments when running the command.

   Example:

```
$ 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

1. Confirm the PingDirectory server is offline.
2. Use the `import-ldif` command to import data from a MakeLDIF template, which is located in the `<server-root>/config/MakeLDIF` directory.

   Do not specify any connection arguments when running the command.

   Example:

   The following example 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

You can run LDIF imports while the server is online from another remote server.

The online import is similar to the offline import, except that you must provide information about how to connect and authenticate to the target server.

To schedule an LDIF file to begin importing at a specific time, use the `--task` and `--start YYYYMMDDhhmmss` options of the `import-ldif` tool.

You can also specify email addresses of users to notify when the import process completes. You can notify them regardless of success or failure or only if the import fails.

Performing an online LDIF import

Steps

1. Confirm the PingDirectory server is online.
2. To import data from an LDIF, use the `import-ldif` command.

   Example:

   ```
   $ 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. Confirm the PingDirectory server is online.
2. To import data from an LDIF file at a scheduled time, use the `import-ldif` command.

   > **Note**
   >
   > To specify a time in the UTC time zone, include a trailing `Z`. Otherwise, the time zone defaults to the time zone configured on the server.

   Example:

```
$ 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
```

Result:

```
Import task 2011102617321510 scheduled to start Oct 26, 2011 1:00:00 AM CDT
```

3. To confirm that you successfully scheduled your import task, use the `manage-tasks` command to view a summary of all tasks on the system.

   Example:

   ```
   $ bin/manage-tasks --summary
   ```

   Result:

   ```
   ID                 Type   Status
   -----------------------------------------------
   2011102617321510 Import Waiting on start time
   ```

4. To monitor the progress of this task, use the `manage-tasks` tool.

   > **Note**
   >
   > 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 PingDirectory server.

   Example:

   ```
   $ bin/manage-tasks --info 2011102617321510
   ```

   Result:

```
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

- To cancel the scheduled task, run the `manage-tasks` tool.

  Example:

  ```
  $ bin/manage-tasks --cancel 2011102417321510
  ```

Adding entries to an existing PingDirectory server

To add entries to an existing PingDirectory server while preserving operational attributes such as `createTimestamp` or `modifiersName`, attach the `Ignore No User Modification` control to the request.

About this task

The `Ignore No User Modification` control allows modification of certain attributes that have the `No User Modification` constraint.

The `Ignore No User Modification` only applies to `ADD` requests. If you use the control to modify an existing entry, it results in an operational attribute change and fail.

To append entries to an existing PingDirectory server, perform the following steps:

Steps

- Run the `ldapmodify` command with the `Ignore No User Modification` control.

  Example:

  The Object Identifier (OID) for the control 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 lets you include or exclude specific attributes and entries during an import.

The `import-ldif` arguments are summarized in the following table.

Inclusion and Exclusion Arguments for import-ldif

Argument	Description
`--includeBranch`	Base distinguished name (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 the server backend to an LDIF file for backups, exporting data to other applications, or reinitializing servers in a replicated topology.

The `export-ldif` tool supports the `--task` argument, which initiates the export process from within the PingDirectory server. When `export-ldif` is run as a task, extra information is available:

- Output from the export process is included in the error log.
- There is a task entry (accessible below the `cn=tasks` base distinguished name (DN)) with information about the success or failure of the export along with log messages.

> **Note**
>
> An `ldif-export` task can be run from the server system or a remote system. Even if it's run from a remote system, the exported LDIF file is still written to the server system.

The `export-ldif` tool exports a point-in-time snapshot of the backend that is guaranteed to provide a consistent state of the database in LDIF. If necessary, you can re-import with `import-ldif`. The data exported by `export-ldif` can include all or some of the entries (a subset of the entries, or a subset of the attributes within entries, or both) contained in the backend. You accomplish this by specifying branches, filters, and attributes to include or exclude. You can compress, encrypt, or digitally sign the exported LDIF.

> **Tip**
>
> You can configure LDIF exports as recurring tasks with `dsconfig create-recurring-task`. You can schedule them to run by adding them to a recurring task chain.

Performing an export

Export data to an LDIF file.

Steps

- To export data to an LDIF file, run the `export-ldif` tool.

  Example:

  ```
  $ bin/export-ldif --backendID userRoot --ldifFile userRoot.ldif
  ```

Performing an export from specific branches

Steps

- To export data to an LDIF file under a specific branch from the userRoot backend of the local PingDirectory server into a compressed file, run the `export-ldif` tool.

  Example:

  In addition to including a specific branch, the following command 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
  ```

Performing post-LDIF-export task processing

When you invoke an LDIF export as an administrative task, the server can perform additional processing after successfully writing the LDIF file.

About this task

Although you can use the Server SDK to develop custom post-LDIF-export task processors, the server also provides the `upload-to-s3` processor type, which you can use to upload the resulting LDIF file to a specified Amazon S3 bucket as a means of an off-site backup mechanism.

To configure the post-LDIF-export upload to the S3 bucket:

Steps

1. If the server isn't configured with an appropriate Amazon AWS external server definition, create one with the necessary settings for connecting and authenticating to the AWS service.

   Example:

   ```
   dsconfig create-external-server \
     --server-name AWS \
     --type amazon-aws \
     --set authentication-method:access-key \
     --set aws-access-key-id:<accessKeyID>  \
     --set aws-secret-access-key:<secretAccessKey>  \
     --set aws-region-name:us-east-1
   ```

2. Configure an instance of the `upload-to-s3` post-LDIF-export task processor with the appropriate settings.

   Example:

   ```
   dsconfig create-post-ldif-export-task-processor \
     --processor-name "Upload to S3" \
     --type upload-to-s3 \
     --set enabled:true \
     --set aws-external-server:AWS \
     --set s3-bucket-name:<bucketName>  \
     --set maximum-file-count-to-retain:20 \
     --set "maximum-file-age-to-retain:1 w"
   ```

3. When performing an LDIF export as an administrative task using the `export-ldif` tool, use the `--postExportProcessor` argument with the `processor-name` value of the desired post-LDIF-export task processor.

   Example:

   ```
   bin/export-ldif \
     --task \
     --hostname ds.example.com \
     --port 636 \
     --useSSL \
     --bindDN uid=admin,dc=example,dc=com \
     --bindPasswordFile /path/to/admin-password.txt \
     --backendID userRoot \
     --ldifFile ldif/userRoot.ldif \
     --postExportProcessor "Upload to S3"
   ```

> **Note**
>
> To specify that a post-LDIF-export task processor should be used when performing an automated LDIF export through a recurring task, set the `post-ldif-export-task-processor` property in the configuration for the recurring task.

Encrypting LDIF exports and signing LDIF files

You can encrypt data during an LDIF export and digitally sign the LDIF file.

The PingDirectory server provides features to encrypt data during an LDAP Data Interchange Format (LDIF) export using the `export-ldif` `--encryptLDIF` option. It also allows the encrypted LDIF file to be imported on the same instance, or another server in the same replication topology, using the `import-ldif` tool. You can use a `--doNotEncrypt` argument 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 can be written to disk.

You can use the `export-ldif` tool with the `--promptForEncryptionPassphrase`, `--encryptionPassphraseFile`, and `--encryptionSettingsDefinitionID` arguments to specify which key to use for encrypting the export. The `import-ldif tool` automatically detects encryption and compression and has `--promptForEncryptionPassphrase`, `--encryptionPassphraseFile` options as well.

The PingDirectory 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**
>
> There is little added benefit to signing and encrypting the same data because encrypted data cannot be altered without destroying the ability to decrypt it.

Encrypting an LDIF export

Steps

- To encrypt the data during an export to an output LDIF file, run the `export-ldif` tool with the `--encryptLDIF` option.

  Example:

  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

Use the `import-ldif` tool to import an encrypted LDIF file.

About this task

You can import an encrypted LDIF file into the same instance from which it was exported or into another server in the same replication topology with that instance. You cannot import an encrypted LDIF file into a server that is not connected to the instance from which it was exported.

Steps

- Run the `import-ldif` tool to import the encrypted LDIF file from Encrypting an LDIF export.

> **Note**
>
> The command imports the `data.ldif` file and 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.

Example:

```
$ bin/import-ldif --backendID userRoot --ldifFile /path/to/data.ldif \
  --overwriteExistingEntries
```

Signing an export

Steps

- To digitally sign the data during an export to an output LDIF file, run the `export-ldif` tool with the `–sign` option.

  Example:

  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

- To import the signed LDIF file (`data.ldif`) from Signing an export, run the `import-ldif` tool with the `--isSigned` option.

  The tool imports the `data.ldif` file and checks the signature of the contents while overwriting the existing contents to the `userRoot` backend.

> **Note**
>
> The command requires the `--isSigned` option, which instructs the tool that the contents of the LDIF file are signed.

Example:

```
$ bin/import-ldif --backendID userRoot \
  --ldifFile /path/to/data.ldif \
  --overwriteExistingEntries --isSigned
```

Filtering data exports

The `export-ldif` command is analogous to the `import-ldif` command to provide a way to include or exclude specific attributes or entries during an export.

Inclusion and Exclusion Arguments for export-ldif

Argument	Description
`--includeBranch`	Base distinguished name (DN) of a branch to include in the LDIF export (can be specified multiple times).
`--excludeBranch`	Base DN of a branch to exclude from the LDIF export (can be specified multiple times).
`--includeAttribute`	Attribute to include in the LDIF export (can be specified multiple times).
`--excludeAttribute`	Attribute to exclude from the LDIF export (can be specified multiple times).
`--includeFilter`	Filter to identify entries to include in the LDIF export (can be specified multiple times).
`--excludeFilter`	Filter to identify entries to exclude from the LDIF export (can be specified multiple times).
`--excludeOperational`	Exclude operational attributes from the LDIF export.
`--excludeReplication`	Exclude replication attributes from the LDIF export.
`--excludeSoftDelete`	Exclude soft delete entries from the LDIF export.

Scrambling data files

The PingDirectory 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 lets you:

- Obscure the values of certain attributes so that it's difficult to determine the original values in the source data.
- Preserve the characteristics of the associated attribute syntax.

This process is repeatable, so if the same value appears multiple times, it yields the same scrambled representation each time. You can apply scrambling to both LDIF entries and LDIF change records.

The process of scrambling data is not the same as encryption. Only use scrambling 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 schema name surrounded by curly braces, such as `{SSHA256}XrgyNdl3fid7KYdhd/Ju47KJQ5PYZqlUlyzxQ28f/QXUnNd9fupj9g==`, the schema name is left unchanged and the rest of the value is 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 schema) 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`. Scrambling Boolean values is not repeatable because the determination to use a value of `TRUE` or `FALSE` is random. 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 relative distinguished name (RDN) components for any attributes to be scrambled.

  For example, if you configure the tool 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 is 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 is over a time range that is double the difference between the time the **`transform-ldif`** tool launches and the timestamp is scrambled. For values where that time difference is less than one day, one day is 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 is nonzero.
- If an attribute has an octet string syntax, it's 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 JavaScript Object Notation (JSON) object, the resulting value is also a JSON object. All field names are left intact, and only the values of those fields can be scrambled. If the `--scrambleJSONField` argument is provided, only the specified fields have values scrambled. Otherwise, the values of all fields are scrambled. Field values are scrambled as follows:
  - Null values aren't scrambled.

- Boolean values are replaced with randomly-selected Boolean values. As with attributes with a Boolean syntax, these values are non-repeatable.
- Number values have only their digits replaced with randomly-selected digits and all other characters (minus sign, decimal point, exponentiation indicator) are left unchanged.
- String values are 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.

Example

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

PingDirectory server provides efficient `backup` and `restore` command line tools that support full backups.

You can schedule backups using either the UNIX-based `cron` scheduler or PingDirectory server's task-based scheduler.

The PingDirectory server can run backups while it is online and processing other requests, so there is no need to shut down the server or place it in read-only mode before starting a backup.

About backing up and restoring data

Administrators should make a comprehensive backup strategy and schedule that consist of daily, weekly, and monthly backups. The plan should include:

- Full backups of the PingDirectory server data, configuration, and backends

- 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**
>
> You can use `dsconfig create-recurring-task` to configure backups as recurring tasks and schedule those tasks as part of a recurring task chain.

If you back up more than one backend, the `backup` tool creates a subdirectory within a specified backup directory for each backend. If you back up only a single backend, then the backup files are placed in the specified directory. A single directory can only contain files from one backend, so 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 distinguished names (DNs) for the backend
- The Java class used to implement the backend logic

The backup descriptor also includes information about the Berkeley DB Java edition version and information about the attribute and virtual list view (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.

You can choose to ignore warnings using the `ignoreCompatibilityWarnings` option to the `restore` tool. Errors always cause the restore operation to fail.

For example, restoring a newer backup into an older version of the server results in a warning. Restoring an older backup into a new version of the server does not result in a warning, but because the `config` and `schema` backends require special handling, the server generates an error if the server versions do not match exactly the major, minor, point, and patch version numbers.

Both the `backup` and `restore` tools provide encryption options that can be used to specify which key to use for encrypting the backup:

- `--promptForEncryptionPassphrase`
- `--encryptionPassphraseFile`
- `--encryptionSettingsDefinitionID`

For backups encrypted with an encryption settings definition or an internal topology key, the server automatically determines the correct key.

Alternately, you can use the `--doNotEncrypt` argument to force a backup to be unencrypted even if automatic encryption is enabled.

If necessary, you can use the `--maxMegabytesPerSecond` argument to impose a limit on the rate at which the backup can be written to disk.

Retaining backups

The backup tool can use the `--retainPreviousFullBackupCount` or `--retainPreviousFullBackupAge` arguments to identify which previous backups to preserve.

Any other backups in that directory are removed. A new backup is always preserved. However, older backups in the same directory are eligible to be removed.

If you include the `--retainPreviousFullBackupCount` argument, that number of the most recent previous full backups are preserved and any other previous full backups are removed. A value of zero can be specified for the `--retainPreviousFullBackupCount` argument so that only the most recent backup is preserved and all previous backups are removed.

If you include the `--retainPreviousFullBackupAge` argument, its value must be a duration represented as an integer followed by a time unit. Any full backups created longer ago than that duration are eligible to be removed.

If you include both the `--retainPreviousFullBackupCount` and `--retainPreviousFullBackupAge` arguments, then only backups that don't satisfy either condition are deleted.

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

Use the `restore` tool to list the backups in a `backup` directory.

Steps

- To list the existing backups in a specific backup directory, run the following command.

```
$ bin/restore --listBackups --backupDirectory  </mybackups>
```

Result:

```
[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 Compressed:     false
Is Encrypted:      false
Has Unsigned Hash: false
Has Signed Hash:   false
```

Backing up all backends

Use `backup` to save the all of the server's backends.

About this task

The `--compress` option can reduce the amount of space that the backup consumes, but can also significantly increase the time required to perform the backup.

Steps

- To back up all backends and compress the backups, run the following.

```
$ bin/backup --backUpAll --compress --backupDirectory  </path/to/backup>
```

Backing up a single backend

Use the `--backendID` argument to specify a single backend to back up.

Steps

- To back up the `userRoot` directory and compress the backup, use the following command.

```
$ bin/backup --backendID userRoot --compress --backupDirectory  </path/to/backup>
```

Performing an offline restore

Use the `restore` command to restore a backed up backend.

About this task

Only a single backend can be restored at a time.

The PingDirectory server must be shut down before performing an offline restore.

> **Note**
>
> The server root directory should never be restored from a file system backup or snapshot.

Steps

1. To stop the PingDirectory server run the following command.

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

2. To restore a saved backup, run the following command.

```
$ bin/restore --backupDirectory  </path/to/backup/userRoot>
```

3. To start the server, run the following command.

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

Assigning an ID to a backup

Use the `--backupID` argument to assign an ID to a backup.

Steps

- To back up the userRoot directory, run the following command.

```
$ bin/backup --backupDirectory  </path/to/backups/>userRoot \
  --backendID userRoot --backupID weekly
```

The `--backupID` argument identifies the backup being created as "weekly".

Result:

The backup file appears under `backups/userRoot` directory as `userRoot-backup-weekly`.

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. Because you can't run recurring tasks, you must run daily operations using cron or another system that can submit the task.

For online (remote) backups, you can conduct the backup operation while the PingDirectory server is online if you provide information about how to connect and to authenticate to the target PingDirectory server.

Steps

- To schedule the backup to occur at a specific time, use the task-based `--start YYYYMMDDhhmmss` option.

> **Note**
>
> To specify a time in the UTC time zone format, add a trailing `Z` to the time. Otherwise, the time is treated as a local time in the time zone configured on the server.

Example:

```
$ bin/backup --backUpAll --task --start 20111025010000 \
  --backupDirectory /path/to/backup --completionNotify admin@example.com \
  --errorNotify admin@example.com
```

Result:

```
Backup task 2011102500084110 scheduled to start Oct 28, 2011 1:00:00 AM CDT
```

Scheduling an online restore

About this task

To perform an online restore, use the Tasks subsystem and enter connection and authentication information and an optional start time.

> **Note**
>
> The server must be online.

The Tasks subsystem allows you to schedule certain operations, such as `import-ldif`, `backup`, `restore`, `start-server`, and `stop-server`.

Steps

- Schedule an online restore.

  Example:

  ```
  $ bin/restore --task --start 20111025010000 \
    --backupDirectory /path/to/backup/userRoot \
    --completionNotify admin@example.com --errorNotify admin@example.com
  ```

  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 can't run recurring tasks, so you must run daily operations using cron or another system that can submit the task.

  > **Note**
  >
  > To specify a time in the UTC time zone, add a trailing `Z` to the time. If you do not specify a time, the configured time zone on the server will be set to a local time.
  > The backend is unavailable while the restore is in progress.

Encrypting a backup

About this task

To encrypt a backup:

Steps

- Go to the server root directory and use the `backup` tool to back up the single backend, `userRoot`, and encrypt it with the `--encrypt` option.

  Example:

  ```
  $ bin/backup --encrypt --backendID userRoot --compress --backupDirectory /path/to/backup
  ```

Signing a hash of the backup

Run the `backup` tool to backup a backend and generate and digitally sign the hash of the backup contents.

Steps

- To backup the single backend from the server root directory, run the `backup` tool.

Choose from:

- To generate and digitally sign the hash of the backup contents, run the `-signHash` option.

```
$ bin/backup --signHash --backupDirectory backups/userRoot --backendID userRoot \
  --backupDirectory /path/to/backup
```

- To only generate a hash of the backup contents, run `backup` with the `--hash` option.

Restoring a backup

You can use the `restore` command to restore a backup created with the `backup` command.

About this task

You can only restore a single backend at a time, and that backend will be offline for the duration of the restore process. The directory containing the backup used for the restoration includes a `backup.info` file with information about the backup, including when the backup was generated and whether the backup is compressed, signed, or encrypted.

You can use the `restore` command for the following data recovery scenarios:

Steps

- To restore a single backend in a replicated topology, run the `dsreplication initialize` command.

  This command streams the contents of the backend database from the source to the destination and ensures that the server receives the most up-to-date version of the data.

- To restore a backup in a non-replicated instance, run the `restore` command and specify the path to the directory containing the backup.

  Example:

```
$ bin/restore --backupDirectory /path/to/backup
```

- To restore a backup to a point-in-time:

  1. Run `dsreplication pre-external-initialization` on a server in the topology.
  2. Choose the required `baseDN`.
  3. Stop the server.
  4. Restore that server to your desired point-in-time backup.
  5. Re-start the server.
  6. Run `dsreplication initialize-all`.
  7. Initialize all other servers in the topology from the restored server.
  8. Run `dsreplication post-external-initialization` on the restored server.

- If all of your servers have been compromised:

    1. Build a directory server.
    2. Restore from the latest backup.
    3. If available, use the `extract-data-recovery-log-changes` command to replay changes in the data recovery log.

       Learn more in Reverting or replaying changes.
    4. Add and initialize new directory servers from the one that you just restored.

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. A plan is the best insurance against significant downtime or data loss in the event of an unrecoverable database issue.

> **Note**
>
> The server root directory should never be restored from a file system backup or snapshot. External backup methods, such as virtual machine (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 PingDirectory server

Take a backup from a local replicated PingDirectory server and restore to the failed server. This is 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 might be preferred over taking a new backup if that would adversely impact performance of the server being backed up although it takes 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.

Using VM snapshots

Although VM snapshots are not recommended for disaster recovery, remember the following in case you must use them:

- Snapshotting the VM should never result in a corrupted PingDirectory server.
- Shutting down the DB before snapshotting is only recommended for speeding up recovery time.

Comparing the data in two PingDirectory servers

The PingDirectory server provides an `ldap-diff` tool to compare the data on two LDAP servers to determine any differences that they might contain.

The differences are identified by first issuing a subtree search on both servers under the base distinguished name (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, such as `"^attrToExclude"`. The `@objectClassName` notation is 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 caused by replication delays. By default, it re-checks each entry twice, pausing two seconds between checks. You can configure these settings with the `--numPasses` and `--secondsBetweenPass` options. If the utility cannot make a clean comparison on an entry, it lists any exceptions in comments in the output file.

The PingDirectory 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 PingDirectory 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 PingDirectory 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. You can customize the memory used by the `ldap-diff` tool 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 PingDirectory servers using ldap-diff

Steps

1. Use `ldap-diff` to compare the entries in two PingDirectory server instances.

   Ignore the `userpassword` attribute because of the one-way password hash used for the password storage scheme.

   Example:

   ```
   $ 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 adds from the perspective of the source server as the authoritative source.

   Example:

```
# 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 PingDirectory server configurations and produce a `dsconfig` batch file needed to align the source with the target.

  Example:

  The following example compares the current configurations of server1 and server2. The changes necessary to align server1's configuration 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
  ```

  You can find more information about runtime options in Available command-line tools or the `config-diff` tool help.

  Important

  The `config-diff` tool doesn't support connecting to different versions over LDAP. To compare configurations between servers of different versions:

  1. Copy the `config/config.ldif` file from the lower-version server to the higher-version server.
  2. Run one of the following commands on the higher-version server:
     - If the higher-version server is the source:

       ```
       $ bin/config-diff --sourceLocal \
         --targetFile "<lower-version config.ldif file>"
       ```

     - If the higher-version server is the target:

       ```
       $ bin/config-diff --targetLocal \
         --sourceFile "<lower-version config.ldif file>"
       ```

Comparing entries using source and target DN files

Steps

- To compare the entries in two PingDirectory server instances, use `ldap-diff`.

  In the following example, the utility uses a single distinguished name (DN) input file for the source and target servers, so that no search filter is used. Ignore the `userpassword` attribute because of the password storage scheme that uses a one-way hashing algorithm.

  Example:

```
$ 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 PingDirectory servers for missing entries only using ldap-diff

Steps

- To compare two PingDirectory servers and return only those entries that are missing on one of the servers, use `ldap-diff` with the `--missingOnly` option.

  This can significantly reduce the runtime for this utility.

  Example:

  ```
  $ 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 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 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, such as by a malicious or defective client, it can 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, such as concurrent database corruption across all instances, it can recover changes that might not otherwise be available.
- Automating the process of identifying changes made in one topology that can be replayed into another topology, such as to replay production changes into an isolated server or topology for testing purposes or to attempt to reproduce a problem.
- 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 is 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 easily fits into the `extract-data-recovery-log-changes` tool's memory.
- The server retains 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 also retains 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 can 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 are converted to a form that reverts 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. This section provides an overview of PingDirectory server group concepts and procedures on setting up and querying groups in the server.

Groups are often used by external clients to control who has access to a particular application or features. Internally, they might be used by the server to control its behavior. 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. 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

Overview of groups

The PingDirectory server provides three types of groups: static, dynamic, and virtual static groups.

A description of each group type follows:

Static groups

A static group is an entry that contains an explicit list of member or uniquemember attributes, depending on its structural object class. Static groups are ideal for small, infrequently changing elements. When the membership list grows, static groups become more difficult to manage because any change in a member base distinguished name (DN) must then be changed in the group. Static groups use one of three structural object classes: `groupOfNames`, `groupOfUniqueNames`, and `groupOfEntries`.

The PingDirectory 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 an LDAP URL. Dynamic groups solve the scalability issues for static groups because searches are efficient, constant-time operations. However, searches that range over a large set of data might affect performance.

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. Avoid using virtual static groups for applications that need to retrieve the entire membership list because the process for constructing the entire membership list can be expensive.

About the isMemberOf and isDirectMemberOf virtual attribute

PingDirectory server can generate either `isMemberOf` or `isDirectMemberOf` virtual attributes in user entries.

The existence of static, nested, dynamic, and virtual static groups can make it complex to work with groups in the server because the ways you interact with them are so different. Static groups can use three different structural object classes, not counting the auxiliary class for virtual static groups, which can further complicate things. PingDirectory server's virtual attributes simplify the group-related determination process for consistency across all types of groups.

The value of the `isMemberOf` virtual attribute is a list of distinguished names (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 `isMemberOf` and `isDirectMemberOf` are operational attributes, only users who have been granted the privilege can see them. 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. You should 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.

The `rewrite-search-filters` `isMemberOf` configuration property

The `rewrite-search-filters` property allows `isMemberOf` searches targeting dynamic groups to be processed more efficiently because you can substitute the filter of the dynamic group into the original search filter. This speeds up paged searches across a large dynamic group.

The following table shows the three possible settings for the `rewrite-search-filters`.

Setting	Description
`always`	Always enhance search filters to include the dynamic group filter components. **Note** This often speeds up searches on dynamic groups, but in rare cases can make search processing slower. An example is if the dynamic group filter matches many entries, but the dynamic group base distinguished name (DN) and scope do not.

Setting	Description
`within-group-scope`	**The default value. Enhances search filters to include dynamic group filter components only when the base DN and scope of the dynamic group.** **Note** Although this doesn't improve performance for search requests whose scope is larger than that of the dynamic group, it should never lead to slower dynamic group search processing
`never`	**Never enhance search filters to include the dynamic group filter components.**

To use the `rewrite-search-filters` configuration property:

- **To set the `rewrite-search-filters` configuration property to the desired setting, run `dsconfig`.**

```
dsconfig set-virtual-attribute-prop --name isMemberOf --set rewrite-search-filters:<always | never |
within-group-scope>
```

- **To optionally keep the server from archiving configurations where only this attribute is changing, you can declare the `rewrite-search-filter` attribute as insignificant by running `dsconfig`.**

```
dsconfig set-backend-prop --backend-name config --add insignificant-config-archive-attribute:ds-cfg-
rewrite-search-filters
```

Determine if a user is a member of a specified group

To determine if a user is a member of a specified group using the `isMemberOf` virtual attribute, 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 base-level search.

Search Parameter	Value
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.

Determine the set of all groups in which a user is a member

To determine the set of all groups in which a user is a member, retrieve the user's entry with a base-level search and include the `isMemberOf` attribute.

Search Parameter	Value
Base DN	`uid=john.doe,ou=People,dc=example,dc=com`
Scope	`base`
Filter	`(objectclass=*)`
Requested attributes	`isMemberOf`

Determine the set of all members for a specified group

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.

Search Parameter	Value
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 previous examples 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**
>
> If this filter targets a dynamic group using an unindexed search, then this might be an expensive operation. However, it is not 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. Otherwise, it would 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 distinguished name (DN)-valued attribute.

There are three types of static groups supported for use in the PingDirectory server: groupOfNames, groupOfUniqueNames, and groupOfEntries. A description for each follows:

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 is in an entry. The PingDirectory 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 is in an entry. The PingDirectory 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 PingDirectory 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

Using an LDIF file, you can configure a static group. Static groups contain a membership list of explicit distinguished names (DNs) specified by the `uniquemember` attribute.

Creating a static group

About this task

To create a static group:

Steps

1. Open a text editor and create a group entry in LDIF.
    1. Include the `groupOfUniquenames` object class and `uniquemember` attributes.
    2. (Optional) If you did not have `ou=groups` set up in your server, add it in the same file.
    3. Save the file.

       Example:

       In the following example, the file is named `static-group.ldif`.

       This 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. To add the group entries to the server, use the `ldapmodify` tool.

   Example:

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

3. To verify the configuration, use the virtual attribute `isDirectMemberOf` that checks membership for a non-nested group.

   The virtual attribute is disabled by default, but you can enable it using `dsconfig`.

   Example:

   ```
   $ bin/dsconfig set-virtual-attribute-prop --name isDirectMemberOf --set enabled:true
   ```

4. To determine if a user is a member of a certain group, use `ldapsearch` to search the `isDirectMemberOf` virtual attribute.

   Example:

   This example inquires if `uid=user.14` is a member of the `cn=Development` group.

   This example assumes that the administrator has the privilege to view operational attributes.

   ```
   $ bin/ldapsearch --baseDN dc=example,dc=com "(uid=user.14)" isDirectMemberOf
   ```

   Result:

```
dn: uid=user.14,ou=People,dc=example,dc=com
isDirectMemberOf: cn=Development,ou=groups,dc=example,dc=com
```

5. Use the group as a target in access control instructions (ACI).

    1. Open a text editor and create an `aci` attribute in an LDIF file.
    2. Save the file.
    3. To add the file, use the `ldapmodify` tool.

       Example:

       In this example, the file is named `dev-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=Development)")
         (version 3.0; acl "Dev Group Permissions";
           allow (write) (groupdn = "ldap:///cn=Development,ou=groups,dc=example,dc=com");)
       ```

       > **Note**
       >
       > You can create a similar ACI for the QA group, which is not shown in the previous example, but is shown in the example for step 1.

6. To add the file, use the `ldapmodify` tool.

   Example:

   ```
   $ 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 new user.

  Example:

  This 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.

  Example:

  This 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 section provides a description of 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

Steps

- To determine if 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 distinguished name (DN) of the specified user.

  > **Note**
  >
  > For best performance, include a specific attribute list, using either `cn`, or a `1.1` request that no attributes be returned, so that the entire member list is not returned.

  Example:

  This table contains the search criteria to determine if 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`".

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`

  Example:

```
$ 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"
```

Result:

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 if the membership attribute for the specified group is `member` or `uniqueMember`, then revise the filter to allow either attribute.

  Example:

  This example adjusts the filter from the previous step's example to expand the membership attribute to allow for `member` and `uniqueMember` attributes.

  ```
  (|(member=uid=john.doe,ou=People,dc=example,dc=com)
  (uniqueMember=uid=john.doe,ou=People,dc=example,dc=com))
  ```

Determining the static groups to which a user belongs

Steps

- 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 directory information tree (DIT).

  > **Tip**
  >
  > Configure the search filter to match any type of static group in which the specified user is a member.

  Example:

  The following table contains the search criteria to determine the set of all static groups in which the user `uid=john.doc,ou=People,dc=example,dc=com` is a member.

Base DN	`dc=example,dc=com`
Scope	`sub`
Filter	`(|(&(objectClass=groupOfNames)(member=uid=john.doe,ou=People,dc=example,dc=com))(&(objectClass=groupOfUniqueNames)(uniqueMember=uid=john.doe,ou=People,dc=example,dc=com))(&(objectClass=groupOfEntries)(member=uid=john.doe,ou=People,dc=example,dc=com)))`
Requested attributes	`1.1`

  Example:

```
$ 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"
```

Result:

Entries returned from the search represent each static group in which the specified user is a member.

> **Note**
>
> A base level search of the user's entry for `isMemberOf` or `isDirectMemberOf` virtual attributes gives the same results. You can also use the virtual attributes with virtual static groups.

Determining the members of a static group

Steps

- To determine all of the members for a static group, retrieve the group entry, including the membership attribute.

> **Tip**
>
> To retrieve attributes from member entries, search all users whose `isMemberOf` attribute contains the group DN, returning the attributes desired.
> To retrieve additional information about the members, such as attributes from member entries, issue a separate search for each member to retrieve the user entry and the desired attributes.

Example:

This table contains the search criteria to retrieve the list of all members for the group `cn=Test Group,ou=Groups,dc=example,dc=com`.

Base DN	`cn=Test Group,ou=Groups,dc=example,dc=com`
Scope	`base`
Filter	`(objectClass=*)`
Requested attributes	`member uniqueMember`

Example:

```
$ bin/ldapsearch --baseDN "cn=Test Group,ou=Groups,dc=example,dc=com" \
  --searchScope base "(objectclass=*)" uniqueMember
```

Result:

The returned entry includes the DNs of all users that are members of that group.

Using inverted static groups

An inverted static group stores its group distinguished name (DN) in the user entries of its members, unlike a traditional static group, which stores the DNs of its members in a `member` attribute (or `uniqueMember` attribute) belonging to the group entry.

Inverted static groups address some of the challenges that can accompany traditional static groups, including:

Scalability

Large traditional static groups mean that you have a large number of members and a corresponding large number of values for the `member` attribute.

- Those large entries are expensive to update, because the server has to rewrite the entire entry for a single change.
- Large entries also require more garbage collection in the database, which puts more pressure on the cleaner.

Nesting

- Traditional static groups can include other groups, and the members of those groups are considered nested members of the outer group. This makes it more complicated to determine whether a user is a member of a group, and the server has to maintain a cache of traditional static group membership to be able to do this quickly, at the cost of additional memory usage.
- The server must do extra processing for changes to group membership, which can be computationally expensive operations.

Why use inverted static groups?

Using inverted static groups enables you to update membership in groups by updating the user entry instead of the group entry. Assuming that individual users aren't members of a large number of groups, you can make these updates without the same loss in performance that you might experience when updating large traditional static groups. For example, adding a user to a group with a million members is as fast as adding a user to a group with a couple of members.

> **Note**
>
> If a user is a member of a large number of inverted static groups, then maintaining a large list of inverted static group memberships in the user's entry could become expensive in the same way as maintaining a large member list for a traditional static group.
> To achieve the best performance in an environment that contains a large number of groups, and that has some large groups and some users that are members of many groups, consider using a mix of traditional and inverted static groups. Use the inverted static groups for groups that have a large number of members and traditional static groups for groups with a small number of members.

Additionally, nested inverted static groups are designed to eliminate the need for caching of inverted static group members, reducing memory pressure and improving server performance. Learn more in Using nesting with inverted static groups.

Creating inverted static groups

You can create an inverted static group to mitigate performance losses associated with a static group that has a large member list.

Before you begin

You must have a parent entry, such as `ou=groups`, set up in your server before adding an inverted static group child entry.

About this task

To create an inverted static group, do the following:

Steps

1. Open a text editor and create a group entry in LDIF:
    1. Include the `ds-inverted-static-group` object class.

    > **Important**
    >
    > You must include the `cn` attribute in an inverted static group entry. The only other non-operational attribute types allowed include:
    > - `description`
    > - `ds-nested-group-dn`
    > - `owner`

    2. Save the file.

    Example:

    This example LDIF file, `inverted-static-group.ldif`, creates one group, `cn=Example Inverted Static Group`.

    ```
    dn: cn=Example Inverted Static Group,ou=groups,dc=example,dc=com
    objectclass: top
    objectclass: ds-inverted-static-group
    cn: Example Inverted Static Group
    ```

2. To add the group entries to the server, use the `ldapmodify` tool.

    Example:

    ```
    $ bin/ldapmodify --defaultAdd --filename inverted-static-group.ldif
    ```

Adding or removing inverted static group members

Steps

- To add a member to the group, in the member entry, add the DN of the inverted static group to the DN of the `ds-member-of-inverted-static-group-dn` attribute.

    Example:

```
dn: uid=gwashington,ou=people,dc=example,dc=com
changetype: modify
add: ds-member-of-inverted-static-group-dn
ds-member-of-inverted-static-group-dn: cn=Example Inverted Static Group,ou=groups,dc=example,dc=com
```

- To remove a member from the group, in the member entry, delete the DN of the inverted static group from the `ds-member-of-inverted-static-group-dn` attribute by changing the previous operation from `add` to `delete`.

Using nesting with inverted static groups

Inverted static groups use a unique attribute for nesting other groups as members within the inverted static group.

Before you begin

You must have the distinguished name (DN) of the nested group that you are adding to the inverted static group. For example, `cn=Group to Nest,ou=groups,dc=example,dc=com`.

About this task

When configuring nesting within a traditional static group, you add both users and groups to the group entry by defining `member` or `uniqueMember` attributes with their DNs. Although this is simple, it doesn't allow you to systematically distinguish between the group's individual user members and the members that are actually nested groups.

Inverted static groups make this distinction by storing the nested group member in a unique attribute. Instead of providing the DN of the parent group to the nested group entry, you provide the nested group's DN to the parent group.

Steps

- To add a nested group to an inverted static group, add the value of the nested group's DN to the `ds-nested-group-dn attribute` in the inverted static group entry.

  Example:

```
dn: cn=Example Inverted Static Group,ou=groups,dc=example,dc=com
changetype: modify
add: ds-nested-group-dn
ds-nested-group-dn: cn=Group to Nest,ou=groups,dc=example,dc=com
```

Searching inverted static groups

Determining membership in inverted static groups is similar to determining membership in traditional static groups and dynamic groups.

Steps

- To query whether a user is a member of a specific inverted static group, you can:

  Choose from:

  - Search for the group DN in the `isMemberOf` attribute of a user.
  - Perform a baseObject search against a user entry with a filter such as `"(isMemberOf=<group_DN>)"`.

> **Note**
>
> If the search returns the user's entry, then the user is a member of that group. If the search doesn't return any entries, then the user isn't a member of the group.

- **Perform a compare operation against a user entry with an attribute type of `isMemberOf` and an assertion value that matches the DN of the target group.**

  > **Note**
  >
  > If the response has a result code of `COMPARE_TRUE`, then the user is a member of the group. If the result code is `COMPARE_FALSE`, then the user is not a member of the group.

- **To identify the entire set of members of an inverted static group, perform a subtree equality search against `isMemberOf`.**

  Example:

  ```
  (isMemberOf=cn=Example Inverted Static Group,ou=groups,dc=example,dc=com)
  ```

Referential integrity for inverted static groups

There are both general and specific plugins that help preserve referential integrity for inverted static groups.

The existing referential integrity plugin also handles inverted static groups and is disabled by default. For more information on enabling this plugin, see Maintaining referential integrity with static groups.

Enabling this plugin ensures the following:

- **Removing an inverted static group removes the corresponding `ds-member-of-inverted-static-group-dn` value from the entries of all members.**
- **Renaming an inverted static group updates the corresponding `ds-member-of-inverted-static-group-dn` value in the entries of all members.**

The inverted static group referential integrity plugin is enabled by default and is designed to:

- **Prevent adding a user to a nonexistent group or a non-inverted static group**
- **Prevent adding a group as a direct member, rather than a nested member, of an inverted static group**
- **Prevent adding a nonexistent entry or non-group as a nested member of an inverted static group**

Using inverted static groups with applications

If an application is configured to work with a traditional static group, and you convert that group to an inverted static group, you can use the Traditional Static Group Support for Inverted Static Groups plugin.

About this task

You might encounter some issues related to differences in the way that inverted static groups and traditional static groups manage membership. The Traditional Static Group Support for Inverted Static Groups plugin, which is disabled by default, provides the following support for treating inverted static groups as if they were traditional static groups:

- Intercepts attempts to add or remove member values and converts them into modify operations of the `ds-member-of-inverted-static-group-dn` attribute in the corresponding user entries.
- Generates a member virtual attribute to handle certain kinds of membership queries.

  The virtual attribute can handle attempts to determine whether the group has a specific member. You can optionally generate the attribute with the entire member list, but this can be computationally expensive, depending on the size of the group.

  > **Note**
  >
  > The virtual attribute only works for compare and baseObject searches. It won't work for subtree searches that try to find all groups in which a user is a member.

Steps

- To enable the Traditional Static Group Support for Inverted Static Groups plugin, enter the following command:

```
dsconfig set-plugin-prop --plugin-name "Traditional Static Group Support for Inverted Static Groups"
--set enabled:true
```

Using dynamic groups

Dynamic groups contain a set of criteria 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 is 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 is no longer considered a member of the dynamic group.

In the PingDirectory 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 is used in the process of making the determination, and any other elements present in the URL are ignored.

Example

For example, the following entry defines a dynamic group in which all users below `dc=example,dc=com` with an `employeeType` value of contractor are 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 fewer than 80,000 entries have the `employeeType` of contractor, you must 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

Configure a dynamic group in the same manner as static groups using an LDIF file.

About this task

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.

Steps

1. Use `ldapsearch` to verify that `uid=user.15` is not part of any group.
    1. Assume that `uid=user.15` is not part of any group.

       You add the user to the dynamic group in a later step.

       Result:

       ```
       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.

       Example:

       ```
       $ bin/ldapsearch --baseDN dc=example,dc=com --searchScope sub "(uid=user.0)" ou isMemberOf
       ```

       Result:

       ```
       dn: uid=user.0,ou=People,dc=example,dc=com
       ou: Engineering
       ```

       Example:

       ```
       $ bin/ldapsearch --baseDN dc=example,dc=com --searchScope sub "(uid=user.15)" ou
       ```

2. Open a text editor, and then create a dynamic group entry in LDIF. Save the file as `add-dynamic-group.ldif`.

   The LDIF defines the dynamic group to include all users who have the `ou=Engineering` attribute.

   Example:

```
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)
```

3. Use `ldapmodify` to add the group entry to the server.

   Example:

   ```
   $ bin/ldapmodify --defaultAdd --filename add-dynamic-group.ldif
   ```

4. 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.

   Example:

   ```
   $ bin/ldapsearch --baseDN dc=example,dc=com "(uid=user.0)" isMemberOf
   ```

   Result:

   ```
   dn: uid=user.0,ou=People,dc=example,dc=com
   isMemberOf: cn=eng-staff,ou=groups,dc=example,dc=com
   ```

5. Run the following command to return the DNs of entries that are part of the `cn=eng-staff` dynamic group and sort them in ascending order by the `sn` attribute.

   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 larger databases, it is not practical to run a database-wide search for all users as there can be a performance hit on the PingDirectory server.

   Example:

   ```
   $ bin/ldapsearch --baseDN dc=example,dc=com --sortOrder sn \
     "(isMemberOf=cn=eng-staff,ou=groups,dc=example,dc=com)" dn
   ```

6. 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 is automatically added to the `eng-staff` dynamic group.

   Example:

   ```
   $ bin/ldapmodify
   dn: uid=user.15,ou=People,dc=example,dc=com
   changetype: modify
   add: ou
   ou: Engineering
   ```

7. Use `ldapsearch` to check if the user is part of the `cn=eng-staff` dynamic group.

Example:

```
$ bin/ldapsearch --baseDN dc=example,dc=com --searchScope sub "(uid=user.15)" isMemberOf
```

Result:

```
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

Steps

- To determine whether a user is a member of a specific dynamic group, run a base-level search to 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.

> **Tip**
>
> 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 is more complicated. While possible, especially in clients 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.

Example:

This table contains the search criteria to determine if the user `uid=john.doe,ou=People,dc=example,dc=com` is a member of the dynamic group with the desired member URL.

Base DN	`uid=john.doe,ou=People,dc=example,dc=com`
Scope	`base`
Filter	`(ou=Engineering)`
Requested Attributes	`1.1`

Example:

```
$ bin/ldapsearch --baseDN "uid=john.doe,ou=People,dc=example,dc=com" \
  --searchScope base "(ou=Engineering)" "1.1"
```

> **Note**
>
> The search requires the user DN to be under the search base defined in the `memberurl` attribute for the user to be a member.

Result:

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

Steps

- **To determine the set of all dynamic groups in which a user is a member, perform a search to find all dynamic group entries defined in the server using a subtree search with a filter of `(objectClass=groupOfURLs)`.**

> **Tip**
>
> 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.

Example:

The following table contains the search criteria to determine the set of all dynamic groups defined in the `dc=example,dc=com` tree.

Base DN	`dc=example,dc=com`
Scope	`sub`
Filter	`(objectClass=groupOfURLs)`
Requested Attributes	`memberURL`

Example:

```
$ bin/ldapsearch --baseDN "dc=example,dc=com" \
  --searchScope sub "(objectClass=groupOfURLs)" "memberURL"
```

Result:

Each entry returned is 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

Steps

- **To determine all members of a dynamic group, issue a search using the base, scope, and filter of the member URL.**

> **Note**
>
> The set of requested attributes should reflect the attributes desired from the member user entries or "1.1" if no attributes are needed.

Example:

This table contains the search criteria to retrieve the list of all members of the group `dc=example,dc=com` with the `cn` and `mail` attributes.

Base DN	`dc=example,dc=com`
Scope	`sub`
Filter	`(employeeType=contractor)`
Requested Attributes	`cn, mail`

Example:

> **Caution**
>
> This search might be extensive if the associated filter is not indexed or if the group contains a large number of members.

```
$ bin/ldapsearch --baseDN "dc=example,dc=com" \
  --searchScope sub "(employeeType=contractor)" "cn, mail"
```

Using dynamic groups for internal operations

Use dynamic groups for internal operations, such as Access control instruction (ACI) or component evaluation.

The PingDirectory server performs the `memberurl` parsing and internal LDAP search. However, the internal search operation cannot be performed with access control instructions 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, as in the following 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

Virtual static groups make it possible to get the efficiency and ease of management of a dynamic group while allowing clients to interact with them as a static group.

About this task

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 PingDirectory server provides a third type of group: a virtual static group, which references another group and provides access to the members of that group as if it was a static group.

Steps

- 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`.

  > **Note**
  >
  > It should include a `ds-target-group-dn` attribute, whose value is the group from which the virtual static group should obtain its members.
  > You must also enable a virtual attribute that allows the `member` attribute to be generated based on membership for the target group.

  Example:

  This example creates 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
  ```

- To enable a configuration object for the virtual static member attribute, use the `set-virtual-attribute-prop` option with `dsconfig`.

  Example:

```
$ bin/dsconfig set-virtual-attribute-prop --name "Virtual Static member" \
  --set enabled:true
```

Note

A configuration object for this virtual attribute is in the server configuration but is disabled by default. If you want to use virtual static groups with the `groupOfUniqueNames` object class, enable the `Virtual Static uniqueMember` virtual attribute in the same way.

Creating virtual static groups

If your application only supports static groups but has scalability issues, using a virtual static group is a possible solution.

About this task

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. Do not use it if an application accesses the full list of group members because of the performance expense at constructing the list.

Note

If you have a small database and an application that requires that the full membership list be returned, 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. In a text editor, create a group entry in LDIF, and then save the file.

   Example:

   In this example, 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.

   In this example, the file is named `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. To add the virtual static group entry to the server, use the **`ldapmodify`** tool.

Example:

```
$ 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. To enable the Virtual Static `uniqueMember` attribute, use the `set-virtual-attribute-prop` option with `dsconfig`.

> **Note**
>
> This attribute is disabled by default.

Example:

```
$ bin/dsconfig set-virtual-attribute-prop --name "Virtual Static uniqueMember" \
  --set enabled:true
```

4. To determine if a user is part of a virtual static group, use `ldapsearch`.

Example:

In Creating dynamic groups, the example sets up `uid=user.0` to be part of the `cn=eng-staff` dynamic group. This example determines if `uid=user.0` is part of the virtual static group using the `isMemberOf` virtual attribute.

```
$ bin/ldapsearch -h server1.example.com -p 389 -D "cn=Directory Manager" \
  -w secret -b dc=example,dc=com" "(uid=user.0)" isMemberOf
```

Example:

The following example determines if `uid=user.0` is part of the virtual static group without using the `isMemberOf` virtual attribute in the `ldapsearch`.

```
$ 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))"
```

Result:

Using the `isMemberOf` virtual attribute returns the following.

```
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
```

> **Note**
>
> You should see the returned `cn=virtualstatic` entry if successful.

5. (Optional) To try searching for a user that is not part of the `cn=eng-staff` dynamic group, use `ldapsearch`.

Example:

This example searches for `uid=user.20`.

```
$ 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))"
```

Result:

No entries are returned.

Searching virtual static groups

Virtual static groups behave like static groups, so 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.

The process for determining all virtual static groups in which a user is a member is similar to the process for standard static groups in which a user is a member. The query provided in the static groups discussion returns virtual static groups in addition to standard 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 retrieve a list of all members of a virtual static group in the same way as a standard static group: 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.

To make the configuration change to allow full membership retrieval, update the virtual attribute used to expose membership.

```
$ bin/dsconfig set-virtual-attribute-prop --name "Virtual Static member" \
  --set allow-retrieving-membership:true
```

> **Note**
>
> This can be an expensive operation, so you should leave the option to allow retrieving virtual static group membership as disabled unless it is required.

Creating nested groups

PingDirectory server supports nested groups where the distinguished name (DN) of an entry that defines a group is included as a member in the parent entry.

About this task

The following example shows a nested static group, such as `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
```

By default, nested group support is enabled on the PingDirectory server.

The PingDirectory server uses a group cache to support nested groups without the performance hit. The cache supports static group nesting that includes other static, virtual static, and dynamic groups. The server provides a new monitoring entry for the group cache, `cn=Group Cache,cn=Monitor`.

In practice, nested groups are not commonly used for the following 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 PingDirectory server-enabled applications that can interact with groups do not provide any support for nesting.

Tip

Disable this support if:

- Nesting support is not needed in your environment.
- 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 might need to make a membership determination.

To create nested static groups:

Steps

1. Open a text editor, and create a group entry in LDIF.
    1. Include the `groupOfUniquenames` object class and `uniquemember` attributes.
    2. If you did not have `ou=groups` set up in your server, then add it in the same file.
    3. Save the file as `nested-group.ldif`.

       Assume that the static groups, `cn=Developers Group` and `cn=QA Group`, have been configured.

       Example:

       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).

```
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. To add the group entry, use `ldapmodify`.

   Example:

   ```
   $ bin/ldapmodify --defaultAdd --filename nested-static-group.ldif
   ```

3. Using the `isMemberOf` virtual attribute that checks the group membership for an entry, verify the configuration.

   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.

   Example:

   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. In a text editor, create an Access control instruction (ACI) in LDIF. Save the file as `eng-group-aci.ldif`.

   > **Tip**
   >
   > Use the group as a target in ACI.

   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=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.

Example:

```
$ 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.

```
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 are returned even when using `isDirectMemberOf` when retrieving the members.

Maintaining referential integrity with static groups

The PingDirectory server automatically updates references to an entry whenever that entry is removed or renamed in the referential integrity process.

About this task

If a user entry is deleted, the referential integrity plugin removes that user from any static groups in which the user was a member. This is not necessary for dynamic groups because no explicit membership is maintained. Similarly, if a modify distinguish name (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.

In the default configuration, the referential integrity processing performs synchronously. Because the necessary cleanup work must complete before the response to the original operation returns, the throughput and response time of delete and modify DN operations might be adversely impacted. To alleviate this performance impact, change the configuration to use a non-zero update interval. As a result, the referential integrity process uses a separate background thread and does not significantly delay the response to delete or modify DN operations.

Performing referential integrity processing in a background thread can introduce a race condition that might adversely impact clients that delete a user and then immediately attempt to re-add it and establish new group memberships. For example, if referential integrity processing has not yet been completed for the delete, then newly-established group memberships might be removed along with those that 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 the client attempts to re-establish those memberships can fail if the referential integrity processing has not removed the previous membership. Use the default synchronous behavior unless the performance impact associated with synchronous referential integrity processing is unacceptable or clients might be affected by delayed referential integrity processing.

Other configuration attributes for this plugin include:

Referential integrity plugin configuration attributes and their descriptions

Configuration attribute	Description
`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. By default, If no values are provided then entries within all public naming contexts will be included.
`log-file`	This attribute specifies the path to a log file that can 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 can 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.

By default, referential integrity support is disabled. To enable the support:

Steps

- **Use the `dsconfig` tool as shown.**

```
$ bin/dsconfig set-plugin-prop --plugin-name "Referential Integrity" \
  --set enabled:true
```

> **Note**
>
> The internal operations of the referential integrity plugin are not replicated. In a replicated topology, you must enable the referential integrity plugin 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 Configuring the Server in PingDirectory Server Administration Guide.

Monitoring the group membership cache

At startup, the PingDirectory server logs information about the group membership cache's memory consumption.

The hard-coded cache contains information about the group memberships for internal processing, such as Access control instructions (ACIs). By default, the group membership cache is enabled.

The information about this cache logs to the standard output log `server.out` and the standard error log. When using groups, use the log information to tune the server for best performance. At startup, the server logs a message to the `server.out` log.

The following is a sample standard output 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 following is a sample error log.

```
[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 entries that are 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.

For most circumstances, use the PingDirectory server without an entry cache. In most cases, the server efficiently retrieves and decode entries from the database. The database cache is more space-efficient than the entry cache, and heavy churn in the entry cache can adversely impact garbage collection behavior.

Entry caches with large static groups

If the PingDirectory server contains large static groups, such as those containing thousands or millions of members, and clients need to frequently retrieve or otherwise interact with these groups, enable an entry cache that holds only large static groups.

In servers containing large static groups, define an entry cache to hold only those large static groups. To define this entry cache, include a filter that matches only group entries, as shown in the following 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 are included. By default, the PingDirectory server entry cache implementation with these settings is disabled. To use this implementation, you must enable these settings.

Enabling the entry cache

Steps

- To enable the entry cache, use the `dsconfig` tool.

  Example:

  ```
  $ bin/dsconfig set-entry-cache-prop --cache-name "Static Group Entry Cache" \
    --set enabled:true
  ```

Creating your own entry cache for large groups

Steps

- To create your own entry cache for large groups, use the `dsconfig create-entry-cache` subcommand.

  Example:

  ```
  # 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

Steps

- To monitor the memory consumed by your entry cache, use the `entry-cache-info` property in the periodic stats logger.
  - To retrieve the monitor entry over LDAP, issue a search on `baseDN="cn=monitor"` using `filter="(objectClass=ds-fifo-entry-cache-monitor-entry)"`.

    > **Note**
    >
    > By default, the entry cache memory is set to 75% with a maximum of 90%.

    Example:

```
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.
```

Tuning the index entry limit for large groups

The PingDirectory server uses indexes to improve database search performance and to provide consistent search rates regardless of the number of database objects stored in the directory information tree (DIT).

About this task

You can specify an index entry limit property that defines the maximum number of entries allowed to match a given index key before the server no longer maintains it.

Steps

- When the index keys have reached the default limit of 4000, rebuild the indexes using the `rebuild-index` tool.

  Example:

  ```
  $ bin/rebuild-index --baseDN dc=example,dc=com --index objectclass
  ```

- For PingDirectory server environment directories containing more than 4000 groups with the same structural object class, such as more than 4000entries, 4000 `groupOfUniqueNames` entries, 4000 `groupOfEntries` entries, or 4000 `groupOfURLs` entries, 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 by setting the `index-entry-limit` property in the command line.

  > **Note**
  >
  > For most PingDirectory server environments, the default index entry limit value of 4000 entries is sufficient. However, group-related processing might be necessary to increase the index entry limit.

  Example:

  ```
  $ bin/dsconfig set-local-db-index-prop --backend-name userRoot \
    --index-name objectClass --set index-entry-limit:175000
  ```

- As an alternative, create a separate backend to hold these group entries so that an unindexed search in that backend yields primarily group entries.

  > **Note**
  >
  > 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 can be expensive.

- For directories in which any single user is be a member of more than 4000 static groups of the same type, increase the index entry limit for the `member` or `uniqueMember` attribute to a value larger than the maximum number of groups in which any user is a member.

  > **Note**
  >
  > If you do not increase the limit, searches to retrieve the set of all static groups in which the user is a member can be unindexed and therefore expensive.

Summary of commands to search for group membership

The following summary of commands show the fastest way to retrieve direct or indirect member distinguished names (DNs) for groups.

- To retrieve direct member, non-nested DNs of group `cn=group.1,ou=groups,dc=example,dc=com`, run the following.

  ```
  $ bin/ldapsearch --baseDN "cn=group.1,ou=Groups,dc=example,dc=com" "(objectClass=*)" uniqueMember member
  ```

- To retrieve direct member entries or non-nested under `dc=example,dc=com` of group `cn=group.1,ou=groups,dc=example,dc=com`, run the following.

  ```
  $ bin/ldapsearch --baseDN "ou=people,dc=example,dc=com" "(isDirectMemberOf=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.

- To retrieve group DNs in which user `uid=user.2,ou=people,dc=example,dc=com` is a direct member or non-nested, static groups, run the following.

```
$ 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`, run the following.

```
$ 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, run the following.

```
$ bin/ldapsearch --baseDN "uid=user.2,ou=people,dc=example,dc=com" "(objectClass=*)" isMemberOf
```

Migrating Oracle 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.

You can migrate Oracle static and dynamic groups to PingDirectory server groups. For more information about the differences in access control evaluation between Oracle and the PingDirectoryserver, see Migrating ACIs from Oracle to the PingDirectory server.

Migrating static groups

About this task

PingDirectory server supports static LDAP groups with structural object classes of `groupOfNames`, `groupOfUniqueNames`, or `groupOfEntries`. In general, you can import static groups without modification.

You can enable a First-In, First-Out (FIFO) entry cache to cache group-to-user mappings, which improves performance when accessing large entries at the expense of greater memory consumption. PingDirectory server provides an out-of-the-box FIFO entry cache object for this purpose. You must enable this object using `dsconfig` as described in Using the entry cache to improve the performance of large static groups.

To migrate static groups:

Steps

1. To enumerate any schema differences between the DSEE deployment and the Ping Identity deployment, use the `migrate-ldap-schema` tool.
2. To enumerate any configuration differences between the DSEE deployment and the Ping Identity deployment, use the `migrate-sun-ds-config` tool.
3. Import or configure any necessary schema or configuration changes recorded by the tools in steps 1 and 2.

4. Import the existing users and groups using the `import-ldif` tool.
5. From the PingDirectory server root directory, open the `docs/sun-ds-compatibility.dsconfig` file using a text editor.
6. Go to the FIFO Entry Cache section.
    1. Read the accompanying comments.
    2. To enable the corresponding `dsconfig` command, delete the comment character ("#").

    Example:

    ```
    $ 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.

    Example:

    ```
    $ bin/dsconfig set-plugin-prop --plugin-name "Referential Integrity" --set enabled:true
    ```

Next steps

If the PingDirectory server is part of a replication topology, enable the Referential Integrity plugin for each replica.

Migrating static groups to virtual static groups

About this task

In many cases, using virtual static groups in place of static groups can produce marked performance gains without having to update client applications. Migrating to virtual static groups varies depending on the original directory information tree (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.

For this task, consider the following:

- The common membership trait for all members of the `All Users` group is the parent distinguish name (DN) `ou=People,dc=example,dc=com`.
- In other cases, a common attribute might need to be used. For example, groups based on the location of its members could use the `l` location or `st` state attribute.
- 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 can have a large membership set without incurring the overhead of a static group.

To migrate Oracle Directory Server Enterprise Edition static groups to virtual static groups:

Steps

1. Create a dynamic group.

Example:

```
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. Create a virtual static group that references the dynamic group.

   Example:

   ```
   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. To populate the `All Users` group with `uniqueMember` virtual attributes, enable the `Virtual Static uniqueMember` virtual attribute.

   Example:

   ```
   $ bin/dsconfig set-virtual-attribute-prop --name "Virtual Static uniqueMember" \
     --set enabled:true
   ```

4. To confirm that the virtual static group is correctly configured, check a user's membership in the group.

   Example:

   ```
   $ 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
   ```

   Result:

   ```
   dn: cn=All Users,ou=Groups,dc=example,dc=com
   ```

5. If a client application requires it, enable the ability to list all members of a virtual static group.

   By default, this feature is disabled.

   Example:

   ```
   $ bin/dsconfig set-virtual-attribute-prop --name "Virtual Static uniqueMember" \
     --set allow-retrieving-membership: true
   ```

> **Tip**
>
> You can implement the virtual static group using the `groupOfNames` object class instead of `groupOfUniqueNames`. To do this, 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, you can import dynamic groups without modification.

To migrate dynamic groups:

Steps

1. To enumerate any schema differences between the Oracle Directory Server Enterprise Edition (DSEE) deployment and the Ping Identity deployment, run the `migrate-ldap-schema` tool.
2. To enumerate any configuration differences between the DSEE deployment and the Ping Identitiy deployment, run the `migrate-sun-ds-config` tool.
3. Import or configure any necessary schema and configuration changes recorded by the tools used in steps 1 and 2.
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 distinguished name (DN) in the PingDirectory 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 distinguished name (DN) in the PingDirectory server. The PingDirectory server automatically creates index files when you initialize a base DN or run the `dsconfig` tool to create a local DB backend. During modify operations, the PingDirectory server updates the database index files.

> **Note**
>
> If you enable encryption on the server, indexes will be encrypted.

The PingDirectory server comes with the following types of indexes:

Default system indexes

Ensures 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

Created for each database suffix. Modify the index to meet your system's requirements using the `dsconfig` tool.

Local DB VLV indexes

Allows a client to request the server to send search results using the virtual list view (VLV) control.

Filtered Indexes

Provides the ability to index an attribute, but only for entries that match a specified filter based on an equality index. You can only use the filtered index for searches containing your specified filter. The filtered index can be maintained independently of the equality index for the attribute even if a normal equality index is not maintained for the attribute.

General tips on indexes

Important critical indexes

The PingDirectory 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
```

> **Tip**
>
> You can define a composite index with a filter pattern of `"(objectClass=?)"` and use it in place of the `objectClass` equality attribute index. The composite index is more performant in cases where many entries have the same object class values.
> In most cases, you should avoid setting the index entry limit too high for the `objectClass` index. In a typical directory of users and groups, you should set the index entry limit for indexes covering the `objectClass` type higher than the total number of groups in the server but lower than the number of users.

Built-in indexes for efficient queries

The PingDirectory 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. You can remove the `mail` and `uid` indexes, but these attributes are referenced from the Password Modify Extended Operation and their removal causes problems with components, such as the Exact Match Identity Mapper. If you remove the `mail` or `uid` indexes, additional configuration changes might be necessary to ensure that the server starts properly. You can safely remove the `cn`, `givenName`, `mail`, `sn`, and `telephoneNumber` indexes if clients do not query on these attributes. This reduces the size of the database both on disk and in memory.

Using composite indexes for performance

When exploded attribute index keys exceed the index entry limit, background cleanup processing can sometimes significantly impact server performance. To avoid this performance impact, in cases where you expect index keys will match a large number of entries, use composite indexes instead of attribute indexes.

Composite indexes don't have the performance and scalability issues that attribute indexes have when working with large ID sets, and they don't require the same expensive cleanup processing for keys that exceed the index entry limit.

For example, you can use a composite index with a filter pattern of `"(givenName=?)"`:

- To replace an attribute index on the `givenName` attribute for the equality and ordering index types
- To process substring searches that contain at least a subInitial (that is, "starts with") component

You can also use a composite index with a filter pattern of `"(cn=?)"` to replace a substring index on the `cn` attribute.

Online rebuilds

When an online index rebuild is in progress, the data in that backend is available and writable although the index being rebuilt is not used. Searches that attempt to use that attribute might be unindexed.

Index rebuild administrative alert

The PingDirectory server generates an administrative alert when the rebuild process begins and ends. It has a degraded-alert-type of `index-rebuild-in-progress` so that a PingDirectoryProxy server can avoid using the PingDirectory 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. For more information, see the list of System indexes.

Indexing certain attributes

Ensure that the following recommendations are used when setting up the indexes:

- Do not use equality and substring indexes for attributes that contain binary data.
- Avoid approximate indexes 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, you should prevent applications from performing unindexed searches so that searches that are not indexed would be rejected rather than tying up a worker thread. Ways to achieve this include the following:

- 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 is only 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.
- You can use the Reject Unindexed Search request control to explicitly indicate that a client does not want the server to process an unindexed search request, regardless of privileges. For information about these controls, see the LDAP SDK. 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 only permits an unindexed search if the request includes the Permit Unindexed Search request control. For more information, see the LDAP SDK and the `ldapsearch` tool.
- 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 are 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 default value of "0" 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 PingDirectory 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 JRE backend index entry limit configuration, such as 4000.

To change the default index entry limit, use the `dsconfig` tool, as 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

Expect a limited amount of database growth because of 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.

Entry IDs

The PingDirectory 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.

Entry IDs can look like the following:

- `id2entry`
- `dn2id`
- `dn2uri`
- `id2children`
- `id2subtree`

Index types

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 and thus has an equality index maintained by the PingDirectory server. The following table describes the index types.

1. PingDirectory server index types

Index Type	Description
Approximate	Used to efficiently locate entries that 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 are exactly equal to a given assertion. Equality indexes can only be applied to attributes that have a corresponding equality matching rule.
Filtered	Uses a defined search filter for a specific attribute. This is an offshoot of the equality index, but you can maintain it independently of the equality index for a specific attribute.
Ordering	Used to efficiently 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 have at least one value for a specified attribute. There is only one presence index key per attribute.

Index Type	Description
Substring	Used to efficiently locate entries that 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 distinguished names (DNs). The `dn2id` database 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 the unique identifiers in its subtree.

Viewing the system indexes

Steps

- To view the system and user indexes, use the **`dbtest`** command.

  Example:

  ```
  $ bin/dbtest list-index-status --baseDN dc=example,dc=com --backendID userRoot
  ```

  The index status indicates whether or not an index is trusted. An untrusted index requires rebuilding.

Managing local DB indexes

To modify the local database (DB) indexes to meet your system's requirements, use the `dsconfig` tool. If you are using the `dsconfig` tool in interactive command-line mode, access the Local DB Index menu from the Basic object menu.

Viewing the list of local DB indexes

Steps

- To view the default list of indexes, run `dsconfig` with the `list-local-db-indexes` option.

  Example:

  ```
  $ bin/dsconfig list-local-db-indexes --backend-name userRoot
  ```

  Result:

  ```
  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

- To view a property assigned set for all local DB indexes, run `dsconfig` with the `--property` option.

  Repeat the option for each property that you want to list.

  Example:

  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, run `dsconfig` with the `get-local-db-index-prop` option and the `--index-name` and `--backend-name` properties.

> **Tip**
>
> To view the advanced properties, add the `--advanced` option to your command.

Example:

```
$ bin/dsconfig get-local-db-index-prop --index-name aci \
  --backend-name userRoot
```

Modifying the configuration of a local DB index

About this task

To modify an index, run the `dsconfig` tool. Any modification or addition of an index requires rebuilding the indexes. In general, an index only needs to be built once after it has been added to the configuration.

> **Note**
>
> If you add an index and then import the data using the **`import-ldif`** tool, the index is automatically rebuilt. If you add an index and then add the data using a different method, you must rebuild the index using the **`rebuild-index`** tool.

Steps

1. To modify an index, run **`dsconfig`** with the `set-local-db-index-prop` option and the `--index-name` and `--backend-name` properties.

   Example:

   This example updates the `prime-index` property, which loads the index at startup. To access this command, use the `--advanced` option.

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

2. To view the index and verify the change, run **`dsconfig`** with the `get-local-db-index-prop` option.

   Example:

   ```
   $ bin/dsconfig get-local-db-index-prop --index-name uid \
     --backend-name userRoot
   ```

3. Stop the server.

   > **Note**
   >
   > Although you can rebuild an index with the server online, it is not recommended.

   Example:

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

4. To rebuild the index, run the `rebuild-index` tool.

   Example:

   ```
   $ bin/rebuild-index --baseDN dc=example,dc=com --index uid
   ```

5. Restart the server.

   Example:

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

Creating a new local DB index

Steps

1. To create a new local DB index, run `dsconfig` with the `--create-local-db-index` option and the `--index-name`, `--backend-name`, and `--set index-type:` <*value*> options.

   Example:

   ```
   $ bin/dsconfig create-local-db-index \
     --index-name roomNumber --backend-name userRoot \
     --set index-type:equality
   ```

2. To view the index, run `dsconfig` with the `get-local-db-index-prop` option.

   Example:

   ```
   $ bin/dsconfig get-local-db-index-prop \
     --index-name roomNumber --backend-name userRoot
   ```

3. Stop the PingDirectory server.

   > **Note**
   >
   > Although you can rebuild an index with the server online, it is not recommended.

   Example:

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

4. To rebuild the index, run the `rebuild-index` tool.

   Example:

```
$ bin/rebuild-index --baseDN dc=example,dc=com --index roomNumber
```

5. Restart the PingDirectory server.

   Example:

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

Deleting a local DB index

About this task

Check that no plugin applications are using the index before deleting it.

> **Note**
>
> When the index is deleted, the corresponding index database is also deleted. The disk space is reclaimed after the cleaner threads begin.

Steps

- To remove an index from the database, run `dsconfig` with the `delete-local-db-index` option.

  Example:

```
$ bin/dsconfig delete-local-db-index \
  --index-name roomNumber --backend-name userRoot
```

Composite indexes

A PingDirectory server composite index can generate from multiple pieces of information, such as a combination of multiple filter components or a combination of filter components and a base distinguished name (DN), or from a single piece of information.

> **Note**
>
> Because composite indexes are much more efficient than exploded indexes when dealing with large numbers of identical keys, use composite indexes for data sets where large numbers of entries (several thousand or more) share the same value for an indexed attribute.

To improve the performance of searches in directories with a large number of entries, especially with a large number of non-leaf entries, use equality composite indexes to combine a mandatory equality filter pattern with an optional base DN pattern.

Equality composite indexes offer two advantages over existing equality attribute indexes in the these types of deployments, a base DN pattern and index pages.

Base DN pattern

If a directory environment has many branches, but searches are often done within specific individual branches, use the base DN pattern to make search processing more efficient. The server only needs to search entries within a target branch.

For example, if the directory contains an `"ou=Customers,dc=example,dc=com"` branch with separate branches below that for sets of customers, such as `"ou=ACME,ou=Customers,dc=example,dc=com"` and `"ou=SHOPCO,ou=Customers,dc=example,dc=com"`, you can define a composite index with a filter pattern of `"(sn=?)"` and a base DN pattern of `"ou=?,ou=Customers,dc=example,dc=com"`. Then use a search with a filter of `"(sn=Smith)"` and a base DN of `"ou=ACME,ou=Customers,dc=example,dc=com"` to narrow the search to the Smiths in the ACME branch.

Index Pages

If many entries have the same value for a specific attribute, you can use composite indexes to break large ID sets up across multiple pages, unlike a 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 returns either:

- One Smith record whose value is a block that contains all 50,000 of the matching entry IDs when the index isn't exploded.

  > **Note**
  >
  > The exploded form is efficient for writing, for example, adding or removing a Smith involves just that one entry ID, but it is expensive for searching because it takes 50,000 reads to get all entry IDs for all of the Smiths.

- 50,000 Smith records that each have a value of the matching entry ID when the index is exploded.

  > **Note**
  >
  > The non-exploded form is efficient for searching because all of the entry IDs return in a single read, but it is expensive for writing. If a Smith must be added or removed, the entire block of 50,000 entry IDs must be rewritten.

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, the composite index returns 10 blocks of 5,000 IDs. 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`. Do not use a equality composite index 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, use an equality attribute index.

Rebuilding indexes

When rebuilding specific non-attribute indexes, it is important to include any appropriate prefixes or postfixes with the index name to avoid errors.

- To rebuild a composite index, use the `rebuild-index` tool. The index name must be preceded by `composite.`

```
bin/rebuild-index --index composite.thisIndexName
```

> **Note**
>
> For the exact name to use in rebuilding the index, run the **`dbtest`** tool with the `list-database-containers` option and `--backendID` property.
>
> ```
> bin/dbtest list-database-containers --backendID {backendID}
> ```

Define the following properties when configuring a composite index.

Composite Index Properties	Description
`index-filter-pattern` (required)	Specifies a single-valued filter property used to identify a portion of the index criteria. You can only specify this at the time that the index definition is created.
`index-base-dn-pattern` (optional)	Specifies a single-valued DN property that indicates that the index is scoped to a specific subtree or subtree pattern. You can only specify this at the time the index definition is created.

Composite index filter patterns

You can use the following filter patterns to define a composite index.

Presence matching

Using a presence matching filter causes the server to index the specified attribute for presence matching, which identifies every entry that contains any value for the target attribute. The server uses this index to process searches with presence components.

Syntax and usage

Presence components use the following syntax: `(attributeName=*)`. You can use a standalone presence component in a simple filter or as part of an AND filter pattern, for example `"(&(objectClass=?)(attributeName=*))"`.

> **Tip**
>
> You can replace existing presence attribute indexes with composite indexes for improved scalability or to limit the scope of index keys by using a base DN pattern.

Equality matching

Using an equality matching filter causes the server to index all unique values for the specified attribute for equality matching. The server uses this index to process searches with:

- Equality filter components that target the specified attribute
- Greater-than-or-equal or less-than-or-equal components that target the specified attribute
- Substring components with a subInitial (starts with) element that targets the specified attribute

Substring filter components can also have subAny (contains) and subFinal (ends with) elements, but the server requires a subInitial element to use a composite index with these types of filters.

Syntax and usage

Equality components use the following syntax: `(attributeName=?)`. You can use an equality component in a simple filter or as part of an AND filter, either with multiple equality components or combined with other supported filter pattern components.

For AND filters that contain multiple equality components:

- You can add an ordering or substring matching component, but you must place it last in the filter.
- All other components in the filter must be equality matching components.

> **Tip**
>
> Learn more about when to use a composite equality index instead of an attribute equality index in Composite indexes.

Static equality matching

Using a static equality filter causes the server to index all entries that have the specified value for the given attribute. The server uses this index to process searches with that specific equality filter component.

Syntax and usage

Static equality components use the following syntax: `(attributeName={staticValue})`. You can use a static equality component in a simple filter or as part of an AND filter, either with multiple static equality components or combined with other supported filter pattern components.

> **Tip**
>
> Use static equality matching to index specific attribute values that are present in a large number of entries.

Substring matching

Using a substring matching filter causes the server to index values for the specified attribute for substring matching with filters that include subAny (contains) or subFinal (ends with) components. The server doesn't use substring matching to index subInitial (starts with) components because they are better handled with equality filter patterns.

The server can use a composite index with this filter pattern to process any searches with substring filter components that target the specified attribute. For filters with subInitial (starts with) elements, the server might prefer to search using an available equality index for the attribute type, if one is available.

Syntax and usage

Substring components use the following syntax: `(attributeName=*?*)`. You can only use one substring component in a given filter pattern, and it needs to be the last component of the pattern.

Approximate matching

Using an approximate matching filter causes the server to index all unique values for the specified attribute for approximate matching. The server uses this index to process searches with approximate matching filter components that target the specified attribute.

Syntax and usage

Approximate matching components use the following syntax: `(attributeName~=?)`. You can use an approximate matching component in a simple filter or as part of an AND filter pattern, for example `"(&(givenName~=?)(sn~=?))"`.

> **Tip**
>
> You can replace existing approximate matching attribute indexes with composite indexes for improved scalability or to limit the scope of index keys by using a base DN pattern.

Guidelines for using AND filter patterns

You can specify an AND filter pattern with one or more presence, wildcard equality, static equality, or wildcard approximate matching filters, followed by a maximum of one wildcard substring filter. Each filter component inside the AND must specify a different attribute type.

Using this filter pattern causes the server to index entries with at least one value for each specified attribute. The server uses this index to process searches using AND filters that contain all but the last of those attribute types in equality filters and that contain the last attribute type in an appropriate filter for the wildcard filter.

For example, the server will use a composite index with filter pattern `"(&(tenantID=?)(sn=?))"` to process AND filters containing a `tenantID` equality filter and either an `sn` equality filter, an `sn` ordering filter, or an `sn` substring filter with at least a subInitial (starts with) element.

JSON indexes

JSON indexing is similar to general attribute indexing. Where a general attribute can be indexed several ways and requires a separate database for each index type, with JSON indexing, there is a single database for each JSON field to use 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. The corresponding database entry value is a 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.

When the JSON field constraint indicates to index a JSON field, JSON indexes are automatically created. Learn more about Configuring JSON attribute constraints.

View indexes 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. Depending on the nature of the search filter, the ID list might contain references to entries that do not actually match the filter, such as if the field is a string and if the filter is configured to use case-sensitive matching.

`equalsAny`

Identifies entries that have 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, but 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:

- Add, delete, modify, and modify distinguish name (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 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 also works with JSON indexes to make it possible to check their validity.
    - When rebuilding a JSON index with the `rebuild-index` tool, the attribute name must be followed by `.json`. Optionally, you can also add `.<fieldName>`, where `<fieldName>` is replaced by the name of the JSON field.

```
bin/rebuild-index --index  <attributeName>.json.<fieldName>
```

> **Note**
>
> For the exact name to use in rebuilding the index, run the **`dbtest`** tool with the `list-database-containers` option and `--backendID` property.
>
> ```
> bin/dbtest list-database-containers --backendID {backendID}
> ```

- 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 database (DB) virtual list view (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 VLV control and the client has been authorized with an access control instruction (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

- To view the default list of indexes, use `dsconfig` with the `list-local-db-vlv-indexes` option.

  Example:

  In this 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.

   > **Note**
   >
   > If you do not set any property values, the default values are assigned.

   Example:

   ```
   $ 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.

> **Note**
>
> You must add the `vlv.` prefix to the index name to rebuild the VLV index.

Example:

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.

   Example:

   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.

   > **Note**
   >
   > You must add the prefix `vlv.` to the index name.

   Example:

   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
   ```

Rebuilding a VLV index

When rebuilding specific non-attribute indexes, it is important to include any appropriate prefixes or postfixes with the index name to avoid errors.

Steps

- When rebuilding a VLV index with the `rebuild-index` tool, the index name must be preceded by `vlv.`

```
bin/rebuild-index --index vlv.thisIndexName
```

> **Note**
>
> For the exact name to use in rebuilding the index, run the **dbtest** tool with the list-database-containers option and --backendID property.
>
> ```
> bin/dbtest list-database-containers --backendID {backendID}
> ```

Deleting a VLV index

About this task

Delete a VLV index using the **dsconfig** tool. Before deleting it, check that the index is not being used in any plugin applications.

Steps

1. To remove a VLV index from the database, use **dsconfig** with the delete-local-db-vlv-index option.

   Example:

   ```
   $ bin/dsconfig delete-local-db-vlv-index --index-name givenName \
     --backend-name userRoot
   ```

2. To verify the deletion, try to view the VLV index.

   Example:

   ```
   $ 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.

About this task

Filtered indexing is primarily useful for cases in which clients frequently issue searches with & filters that meet the following criteria:

- The & filter itself matches a small number of entries, but each of the individual components can match a 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 does not change.
- The filter must be narrowed to a base distinguished name (DN) for data structures with many branches or if indexed attribute values appear in a large number of entries.

For an example, consider the following use case process. 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 can exceed the index entry limit because there are many companies. Also, 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.

When configuring a filtered index, define the following properties.

Filtered Index Properties and their descriptions

Filtered Index Properties	Description
`equality-index-filter`	Specifies a search filter that can 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 is maintained for the associated attribute, but only for entries that match the provided filter. The index is used only for searches containing an equality component with the associated attribute type used 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 does not indexed for equality by itself but only in conjunction with the defined equality index filters.

Steps

- Define the `equality-index-filter` filter index property.
- Define the `maintain-equality-index-without-filter` filter index property.

> **Note**
>
> Maintain the filtered index independently from the equality filter for that attribute. Use the filtered index only for searches containing the equality component with the associated attribute type used 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.

Result:

The searches are configured and built with the `rebuild-index` tool or `import-ldif` tool.

This example shows the result of searches built with filters based on the use case process described above with the following 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)))
```

Creating a filtered index

Steps

1. To create a filtered index, use the `dsconfig` tool.

   The following command creates an equality index on the `companyDomain` attribute and maintains an index for the equality filter defined `"(objectclass=company)"`.

   Example:

   ```
   $ 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. After you have created the index, rebuild the indexes.

   1. Stop the PingDirectory server using `bin/stop-server`.
   2. Run the `rebuild-index` tool.

      Example:

      ```
      $ bin/rebuild-index --baseDN dc=example,dc=com --index companyDomain
      ```

3. Start the PingDirectory 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.

To process high write load operations, the server supports an optional expanded index database using an exploded format. To view the current state of the server's indexes and make adjustments to the index databases, the PingDirectory server automatically generates an Index Summary Statistics table after each LDIF import or index rebuild. To determine the key data size for the indexes, the `dbtest` tool includes an index histogram. This section describes 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.

> **Note**
>
> Because composite indexes are much more efficient than exploded indexes when dealing with large numbers of identical keys, use composite indexes for data sets where large numbers of entries (several thousand or more) share the same value for an indexed attribute.

If any index keys have already reached this limit, you must rebuild these indexes before they can use a new limit. If you configure `index-entry-limit` to be greater than 50000, any keys with more than 50000 entries are stored in a separate database in an expanded, or exploded, format.

All keys with less than 50000 entries continue to be stored in one database in a consolidated format so that changes to the key require rewriting all the entry IDs matching the key. All keys with more than 50000 entries and less than the `index-entry-limit` are stored in a separate database in an exploded format so that changes to the key require writing to the updated entry ID only.

If you increase the `index-entry-limit` 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 is stored in a separate database where each key is stored with its entry ID individually. The consolidated format is efficient for read operations because the server can retrieve a row of entry IDs at once, while the exploded format is efficient for high volumes of write operations because it avoids large on-disk growth.

About monitoring index entry limits

Index keys that have reached their limit require indexes to be rebuilt before they can use a new limit. To avoid a potentially costly rebuild, there are several ways to monitor index limits.

In certain cases, it's 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 applications frequently need to find entries of a specific type. However, the top objectClass key doesn't need to be indexed because it appears in every entry in the server.

Select 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 `--listKeysNearestIndexEntryLimt` argument of the `verify-index` tool lists a specified number of keys that are closest to the limit without having exceeded it. Make the index entry limit 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 you need to adjust the index entry limit needs to be adjusted.

The `--listKeysExceedingIndexEntryLimit` argument of the `verify-index` tool lists 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 been exceeded 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. Then, run the `rebuild-index` tool or export to LDIF and re-import.

The server provides other methods for determining if index keys have exceeded or are close to exceeding the index entry limit, including:

- 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.
  - The number of keys for which the number of matching entries falls within several predefined buckets, such as 1–4 entries, 5–9 entries, 10–99 entries, and 100–999 entries.
- During a search operation, if the server accesses one or more index keys whose values have exceeded the index entry limit, the access log message for that operation includes an `indexesWithKeysAccessedExceedingEntryLimit` field containing a comma-delimited list of the appropriate indexes. The same access log field can 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.
- During a search operation, if the server accesses one or more index keys whose values have not yet exceeded the index entry limit but are greater than 80 percent of it, the access log message for that operation includes an `indexesWithKeysAccessedNearEntryLimit` field containing a comma-delimited list of the appropriate indexes. The same access log field can 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%of the index entry limit.
- If a search operation request includes either the `debugsearchindex` attribute or the matching entry count request control with debugging enabled, the debug information includes any indexes accessed that have exceeded the index entry limit or that are within 80% 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 several 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 because of 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% 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% 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 following information:

- Type
- Entry count (the number of records in the database)
- Index status
  - `TRUSTED` indicates the indexes are up-to-date.
  - `UNTRUSTED` indicates the index needs rebuilding.
- Each key's total data size
- Each key's average data size
- Each key's maximum data size

> **Note**
>
> Any indexes that are in exploded format are also listed in 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
```

dbtest output including index databases

Configuring the index properties

By default, the `index-entry-limit` value is 4000, meaning the server stops maintaining index values for keys that match more than 4000 entries. You can change the value with the `dsconfig` tool.

About this task

Before running the following commands, be aware that you must 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` with the `set-backend-prop` subcommand.
2. Set the `index-entry-limit` to 5000. Confirm that you want to apply the changes.

   By default, this value is set to 4000.

   > **Note**
   >
   > Remember to include the bind parameters for your system.

   Example:

   ```
   $ 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
   ```

3. Stop the server.

   Example:

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

4. Rebuild the index.

   Example:

   ```
   $ bin/rebuild-index --baseDN dc=example,dc=com \
     --index cn --index givenName --index objectClass \
     --index sn --maxThreads 10
   ```

5. View the Index Summary Statistics table, which automatically displays to system out after running the `rebuild-index` command.

You can also access the table at `logs/tools/rebuild-index-summary.txt`.

Example:

```
          --- 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
```

6. Repeat steps 1-4 to make more adjustments to your indexes.
7. Restart the server.

Example:

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

About the Index Summary Statistics table

The PingDirectory server automatically generates an Index Summary Statistics table, which you can use to determine the optimal configuration for your system's indexes.

The table only generates when you run the `rebuild-index` tool in offline mode. The table generates after any LDIF import or an index rebuild. It is written to system out and to `logs/tools/rebuild-index-summary.txt`.

The table displays the following information:

- The current index entry limit set by the `index-entry-limit` property on the local database configuration
- The number of keys whose entry count exceeds this limit, if any
- The maximum entry count for any key in the index
- A histogram of the number of keys whose entry falls within a range of values

The following image is an example of the Index Summary Statistics table for the `sn.equality` and the `sn.substring` indexes.

In this figure, 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
```

Example of an Index Summary Statistics Table

Managing entries

The PingDirectory 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.

The following sections describe how to work with entries:

Searching entries

The PingDirectory server provides an `ldapsearch` tool to search for entries or attributes within your server.

The `ldapsearch` tool requires the LDAP connection parameters 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

Searches only for the specified entry.

one

Searches only the children of the entry and not the entry itself.

sub

Searches the entry and its descendants.

The `ldapsearch` tool provides basic functionality specified by the RFC 2254 and provides additional features that use the PingDirectory server's control mechanisms.

> **Note**
>
> For more information, run the **`ldapsearch --help`** function.

Searching the root DSE

The root DSA-specific entry (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.

About this task

Specifically, the root DSE entry provides information about the supported LDAP3 controls, Simple Authentication and Security Layer (SASL) mechanisms, password authentication schemes, supported LDAP protocols, additional features, naming contexts, extended operations, and server information.

The PingDirectory 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. After you set this property, the associated attribute types are recreated and re-registered as user attributes in the schema (in memory, not on disk). After you set the property, you can use `ldapsearch` without "+" to view the root DSE.

Steps

1. To set the `show-all-attributes` property to TRUE, run the `dsconfig` tool.

   Example:

   ```
   $ bin/dsconfig set-root-dse-backend-prop --set show-all-attributes:true
   ```

2. Run the `ldapsearch` tool to view the root DSE entry on the PingDirectory server.

   > **Note**
   >
   > To display the operational attributes in the entry, use `+`.

   Example:

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

Searching all entries in the PingDirectory server

Steps

- To search all entries in the PingDirectory server, run the `ldapsearch` tool.

  > **Note**
  >
  > The filter `"(objectclass=*)"` matches all entries. If you do not specify the `--searchScope` subcommand, it defaults to a search scope of `sub`.

  Example:

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

Searching for an access control instruction

Steps

- To search the `dc=example,dc=com` base distinguished name (DN) entry, run the `ldapsearch` tool.

> **Note**
>
> 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 access control instruction (ACI).

Example:

```
$ bin/ldapsearch --baseDN dc=example,dc=com "(aci=*)" aci
```

Result:

The system displays the following ACI information.

```
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

- To search the `cn=schema` entry, run the `ldapsearch` tool.

> **Note**
>
> The base distinguished name (DN) is specified as `cn=schema` , and the filter `"(objectclass=*)"` matches all entries. The command uses a special attribute "+" to return all operational attributes.

Example:

```
$ bin/ldapsearch --baseDN cn=schema \
  --searchScope base "(objectclass=*)" "+"
```

Searching for a single entry using base scope and base DN

Steps

- To search for a single entry, run the `ldapsearch` tool and specify the base scope and DN.

Example:

```
$ 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

- To search for a single entry, specify the `sub` scope and a search filter that describes a single entry.

  Example:

  ```
  $ bin/ldapsearch --baseDN ou=People,dc=example,dc=com \
    --searchScope sub "(uid=user.14)"
  ```

Searching for all immediate children for restricted return values

Steps

- To search for all immediate children of restricted return values, run the `ldapsearch` tool.

  > **Note**
  >
  > The special attribute "+" returns all operational attributes.

  Example:

  In this sample, the immediate children are `ou=People,dc=example,dc=com`, and the restricted returned values are `sn` and `givenName`.

  ```
  $ 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

- To search for all children of an entry in the appropriate subtree and sort the results by one or more attributes, run the `ldapsearch` command and include the `--sortOrder` option.

  Choose from:

  - To sort the results in ascending order, supply an attribute name for the value of `--sortOrder`.
  - To sort the results in descending order, supply an attribute name preceded by a `-` for the value of `--sortOrder`.
  - To sort the results by multiple attributes, supply the attribute names for the value of `--sortOrder` as follows:
    - List the attributes in the order in which you want to sort the results.

- Separate the attributes with commas.
- Remember to prepend `-` to any attributes that represent a descending sort.

Example

For this example, the `ldapsearch` command searches for children in the `ou=People,dc=example,dc=com` subtree and sorts them first by `sn` (ascending) and then by `givenName` (descending):

```
$ 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

- To search for a subset of the entries in the `ou=People,dc=example,dc=com` subtree, specify a compound filter.

  Example:

  In the following example, 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
```

Optimize paged searches using caching

When the server processes an indexed search using the simple paged results control, it builds the entire candidate set of entry IDs for entries that match the search before serving a subset of entries on a results page. By default, the server does this for every page. The larger the candidate set, the longer the search processing takes.

To improve paged search speed, you can configure the server to cache the generated candidate set using the `simple-paged-results-id-set-cache-duration` property. For example:

```
$ bin/dsconfig set-backend-prop \
  --backend-name userRoot \
  --set "simple-paged-results-id-set-cache-duration:1 m"
```

With this property configured, although the server still has to build the entire candidate set before serving the first page of results, it can cache and reuse that set when returning subsequent pages.

By default, the cache duration is set to 0 milliseconds, and no caching is performed. If you change the value to a non-zero duration, the server caches and reuses the candidate set for the subsequent pages of results, up to the point where the duration between requests for the next page of results exceeds the cache duration.

You should configure a duration that is long enough to ensure that the cached result set doesn't expire before legitimate clients can make use of it, although making the duration too long can cause the server to hold onto the cached result sets longer than necessary, increasing the amount of memory required to hold them.

In most cases, we recommend setting a duration of 30–60 seconds.

When the cache is enabled, it's not shared between server instances. If possible, clients should send requests for all pages of a paged search to the same server, whether caching is enabled or not. The PingDirectoryProxy server automatically tries to route requests to the same backend server for paged searches, but environments that use other forms of load balancing might not conform to this best practice by default. PingDirectory can properly handle cases where client requests for different pages are sent to different servers, although each server instance involved builds the paged search request candidate set independently.

> **Note**
>
> When optimizing paged searches, you should make the previous configuration change for each local database backend where you expect searches with paged results and enable it for all servers in the topology that might process searches using the paged results request control.
> Only enable simple paged results ID caching in environments where all servers support this feature. In a mixed-version topology that includes older versions of the PingDirectory server that don't support paged results caching, those servers won't be able to decode paged results cookies generated by newer servers that do support caching. If you disable paged results caching on the newer servers, the cookies these servers generate will be compatible with the older servers.

Getting information about how indexes are used in a search operation

Before you begin

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).

Steps

- To search for how indexes are used in a search operation, you can either:

  Choose from:

  - Issue a search request with the desired base distinguished name (DN), scope, and filter, and request that the server return the special `debugsearchindex` attribute using the `ldapsearch` command.

```
$ 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)
```

Note

Users must have 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\";)"
```

- **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` using `ldapsearch`.**

```
$ 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
```

Localization of searches with collation matching

You can use collation matching rules to perform locale-specific searches for entries containing non-ASCII data.

Standard LDAP matching rules used for string-type attributes are typically designed for ASCII data. Although PingDirectory can search for attributes that contain non-ASCII data, the default matching rules have some limitations, including:

- They evaluate diacritical marks such as accents and umlauts when comparing characters.
  - The filter `"(givenName=Francois)"` doesn't match a user whose `givenName` value is François.
  - The filter `"(givenName=François)"` doesn't match a user whose `givenName` value is Francois.

- They don't account for different encoding methods for the same character. The Unicode specification makes it possible to encode some non-ASCII characters in multiple ways, but the default matching rules might only match values that use the same encoding as used in the filter.

These limitations can affect searches related to non-ASCII data that reflects a regional language variant, referred to as a locale. When searching for this type of data, you can tell the server to search using a collation matching rule specific to the locale.

To search with a collation matching rule, use an extensible matching filter with the IDs for the collation matching rule and the locale. Include these values by either specifying the locale abbreviation followed by the rule abbreviation or the locale OID followed by the OID suffix of the rule.

For example, to search for entries with a `givenName` value that matches "Céline" for the French Canadian locale, you could use either of the following filters:

- `(givenName:fr-CA.eq:=Céline)`
- `(givenName:1.3.6.1.4.1.42.2.27.9.4.78.1.3:=Céline)`

> **Note**
>
> PingDirectory doesn't support indexing extensible matching filters. The server treats searches involving collation matching rules as unindexed unless they are included in an `AND` filter that includes other components that can be indexed.

Collation matching rules

The following table lists the supported collation matching rules, correlated with their abbreviations and OID suffixes:

Rule	Description	Rule abbreviations	OID suffix
Equality	The filter matches entries that have the specified value for the target attribute in accordance with the associated locale. **Note** For locale-specific filters, you can omit the abbreviation or OID suffix for the equality rule.	`.eq`	`.3`
Less-than	The filter matches entries that have a value for the target attribute that comes before the specified value when using lexicographic ordering, in accordance with the associated locale.	`.lt`	`.1`

Rule	Description	Rule abbreviations	OID suffix
Less-than-or-equal-to	The filter matches entries that have a value for the target attribute that either matches the specified value or comes before the specified value when using lexicographic ordering, in accordance with the associated locale.	`.le` or `.lte`	`.2`
Greater-than	The filter matches entries that have a value for the target attribute that comes after the specified value when using lexicographic ordering, in accordance with the associated locale.	`.gt`	`.5`
Greater-than-or-equal-to	The filter matches entries that have a value for the target attribute that either matches the specified value or comes after the specified value when using lexicographic ordering, in accordance with the associated locale.	`.ge` or `.gte`	`.4`

Locale language codes

The following table lists the languages by locale that PingDirectory supports for collation matching searches, correlated with their abbreviations and OIDs.

> **Important**
>
> PingDirectory can only support collation matching rules for a given locale if the underlying JVM supports the locale. Check the `matchingRules` attribute of the `cn=schema` entry to determine whether the server supports collation matching for your desired locale.

Language by locale	Locale abbreviation	OID
Afrikaans	af	Step 1.3.6.1.4.1.42.2.27.9.4.1.1
Albanian, Albanian (Albania)	sq, sq-AL	Step 1.3.6.1.4.1.42.2.27.9.4.127.1
Amharic	am	Step 1.3.6.1.4.1.42.2.27.9.4.2.1

Language by locale	Locale abbreviation	OID
Arabic	ar	Step 1.3.6.1.4.1.42.2.27.9.4.3.1
Arabic (Algeria)	ar-DZ	Step 1.3.6.1.4.1.42.2.27.9.4.6.1
Arabic (Bahrain)	ar-BH	Step 1.3.6.1.4.1.42.2.27.9.4.5.1
Arabic (Egypt)	ar-EG	Step 1.3.6.1.4.1.42.2.27.9.4.7.1
Arabic (India)	ar-IN	Step 1.3.6.1.4.1.42.2.27.9.4.8.1
Arabic (Iraq)	ar-IQ	Step 1.3.6.1.4.1.42.2.27.9.4.9.1
Arabic (Jordan)	ar-JO	Step 1.3.6.1.4.1.42.2.27.9.4.10.1
Arabic (Kuwait)	ar-KW	Step 1.3.6.1.4.1.42.2.27.9.4.11.1
Arabic (Lebanon)	ar-LB	Step 1.3.6.1.4.1.42.2.27.9.4.12.1
Arabic (Libya)	ar-LY	Step 1.3.6.1.4.1.42.2.27.9.4.13.1
Arabic (Morocco)	ar-MA	Step 1.3.6.1.4.1.42.2.27.9.4.14.1
Arabic (Oman)	ar-OM	Step 1.3.6.1.4.1.42.2.27.9.4.15.1
Arabic (Qatar)	ar-QA	Step 1.3.6.1.4.1.42.2.27.9.4.16.1
Arabic (Saudi Arabia)	ar-SA	Step 1.3.6.1.4.1.42.2.27.9.4.17.1
Arabic (Sudan)	ar-SD	Step 1.3.6.1.4.1.42.2.27.9.4.18.1
Arabic (Syria)	ar-SY	Step 1.3.6.1.4.1.42.2.27.9.4.19.1
Arabic (Tunisia)	ar-TN	Step 1.3.6.1.4.1.42.2.27.9.4.20.1
Arabic (United Arab Emirates)	ar-AE	Step 1.3.6.1.4.1.42.2.27.9.4.4.1
Arabic (Yemen)	ar-YE	Step 1.3.6.1.4.1.42.2.27.9.4.21.1
Armenian	hy	Step 1.3.6.1.4.1.42.2.27.9.4.89.1
Bangla	bn	Step 1.3.6.1.4.1.42.2.27.9.4.24.1
Basque	eu	Step 1.3.6.1.4.1.42.2.27.9.4.70.1
Belarusian, Belarusian (Belarus)	be, be-BY	Step 1.3.6.1.4.1.42.2.27.9.4.22.1
Bulgarian, Bulgarian (Bulgaria)	bg, bg-BG	Step 1.3.6.1.4.1.42.2.27.9.4.23.1

Language by locale	Locale abbreviation	OID
Catalan	ca, ca-ES	Step 1.3.6.1.4.1.42.2.27.9.4.25.1
Chinese	zh	Step 1.3.6.1.4.1.42.2.27.9.4.143.1
Chinese (China)	zh-CN	Step 1.3.6.1.4.1.42.2.27.9.4.144.1
Chinese (Hong Kong)	zh-HK	Step 1.3.6.1.4.1.42.2.27.9.4.145.1
Chinese (Mongolia)	zh-MO	Step 1.3.6.1.4.1.42.2.27.9.4.146.1
Chinese (Singapore)	zh-SG	Step 1.3.6.1.4.1.42.2.27.9.4.147.1
Chinese (Taiwan)	zh-TW	Step 1.3.6.1.4.1.42.2.27.9.4.148.1
Cornish	kw	Step 1.3.6.1.4.1.42.2.27.9.4.99.1
Croatian, Croatian (Croatia)	hr, hr-HR	Step 1.3.6.1.4.1.42.2.27.9.4.87.1
Czech, Czech (Czech Republic)	cs, cs-CZ	Step 1.3.6.1.4.1.42.2.27.9.4.26.1
Danish, Danish (Denmark)	da, da-DK	Step 1.3.6.1.4.1.42.2.27.9.4.27.1
Dutch, Dutch (Netherlands)	nl, nl-NL	Step 1.3.6.1.4.1.42.2.27.9.4.105.1
Dutch (Belgium)	nl-BE	Step 1.3.6.1.4.1.42.2.27.9.4.106.1
English, English (United States)	en, en-US	Step 1.3.6.1.4.1.42.2.27.9.4.34.1
English (Australia)	en-AU	Step 1.3.6.1.4.1.42.2.27.9.4.35.1
English (Canada)	en-CA	Step 1.3.6.1.4.1.42.2.27.9.4.36.1
English (Hong Kong)	en-HK	Step 1.3.6.1.4.1.42.2.27.9.4.38.1
English (India)	en-IN	Step 1.3.6.1.4.1.42.2.27.9.4.40.1
English (Ireland)	en-IE	Step 1.3.6.1.4.1.42.2.27.9.4.39.1
English (Malta)	en-MT	Step 1.3.6.1.4.1.42.2.27.9.4.41.1
English (New Zealand)	en-NZ	Step 1.3.6.1.4.1.42.2.27.9.4.42.1
English (Philippines)	en-PH	Step 1.3.6.1.4.1.42.2.27.9.4.43.1
English (Singapore)	en-SG	Step 1.3.6.1.4.1.42.2.27.9.4.44.1
English (South Africa)	en-ZA	Step 1.3.6.1.4.1.42.2.27.9.4.46.1

Language by locale	Locale abbreviation	OID
English (U.S. Virgin Islands)	en-VI	Step 1.3.6.1.4.1.42.2.27.9.4.45.1
English (United Kingdom)	en-GB	Step 1.3.6.1.4.1.42.2.27.9.4.37.1
English (Zimbabwe)	en-ZW	Step 1.3.6.1.4.1.42.2.27.9.4.47.1
Esperanto	eo	Step 1.3.6.1.4.1.42.2.27.9.4.48.1
Estonian, Estonian (Estonia)	et, et-EE	Step 1.3.6.1.4.1.42.2.27.9.4.69.1
Faroese	fo	Step 1.3.6.1.4.1.42.2.27.9.4.75.1
Finnish, Finnish (Finland)	fi, fi-FI	Step 1.3.6.1.4.1.42.2.27.9.4.74.1
French, French (France)	fr, fr-FR	Step 1.3.6.1.4.1.42.2.27.9.4.76.1
French (Belgium)	fr-BE	Step 1.3.6.1.4.1.42.2.27.9.4.77.1
French (Canada)	fr-CA	Step 1.3.6.1.4.1.42.2.27.9.4.78.1
French (Luxembourg)	fr-LU	Step 1.3.6.1.4.1.42.2.27.9.4.80.1
French (Switzerland)	fr-CH	Step 1.3.6.1.4.1.42.2.27.9.4.79.1
Galician	gl	Step 1.3.6.1.4.1.42.2.27.9.4.82.1
German, German (Germany)	de, de-DE	Step 1.3.6.1.4.1.42.2.27.9.4.28.1
German (Austria)	de-AT	Step 1.3.6.1.4.1.42.2.27.9.4.29.1
German (Belgium)	de-BE	Step 1.3.6.1.4.1.42.2.27.9.4.30.1
German (Luxembourg)	de-LU	Step 1.3.6.1.4.1.42.2.27.9.4.32.1
German (Switzerland)	de-CH	Step 1.3.6.1.4.1.42.2.27.9.4.31.1
Greek, Greek (Greece)	el, el-GR	Step 1.3.6.1.4.1.42.2.27.9.4.33.1
Gujarati	gu	Step 1.3.6.1.4.1.42.2.27.9.4.83.1
Hebrew, Hebrew (Israel)	he, he-IL	Step 1.3.6.1.4.1.42.2.27.9.4.85.1
Hindi, Hindi (India)	hi, hi-IN	Step 1.3.6.1.4.1.42.2.27.9.4.86.1
Hungarian, Hungarian (Hungary)	hu, hu-HU	Step 1.3.6.1.4.1.42.2.27.9.4.88.1
Icelandic, Icelandic (Iceland)	is, is-IS	Step 1.3.6.1.4.1.42.2.27.9.4.91.1

Language by locale	Locale abbreviation	OID
Indonesian, Indonesian (Indonesia)	id, id-ID	Step 1.3.6.1.4.1.42.2.27.9.4.90.1
Irish, Irish (Ireland)	ga, ga-IE	Step 1.3.6.1.4.1.42.2.27.9.4.81.1
Italian, Italian (Italy)	it, it-IT	Step 1.3.6.1.4.1.42.2.27.9.4.92.1
Italian (Switzerland)	it-CH	Step 1.3.6.1.4.1.42.2.27.9.4.93.1
Japanese, Japanese (Japan)	ja, ja-JP	Step 1.3.6.1.4.1.42.2.27.9.4.94.1
Kalaallisut	kl	Step 1.3.6.1.4.1.42.2.27.9.4.95.1
Kannada	kn	Step 1.3.6.1.4.1.42.2.27.9.4.96.1
Konkani	kok	Step 1.3.6.1.4.1.42.2.27.9.4.98.1
Korean, Korean (South Korea)	ko, ko-KR	Step 1.3.6.1.4.1.42.2.27.9.4.97.1
Latvian, Latvian (Latvia)	lv, lv-LV	Step 1.3.6.1.4.1.42.2.27.9.4.101.1
Lithuanian, Lithuanian (Lithuania)	lt, lt-LT	Step 1.3.6.1.4.1.42.2.27.9.4.100.1
Macedonian, Macedonian (North Macedonia)	mk, mk-MK	Step 1.3.6.1.4.1.42.2.27.9.4.102.1
Maltese, Maltese (Malta)	mt, mt-MT	Step 1.3.6.1.4.1.42.2.27.9.4.104.1
Manx Gaelic	gv	Step 1.3.6.1.4.1.42.2.27.9.4.84.1
Marathi	mr	Step 1.3.6.1.4.1.42.2.27.9.4.103.1
Norwegian, Norwegian (Norway)	no, no-NO	Step 1.3.6.1.4.1.42.2.27.9.4.107.1
Norwegian (Bokmål)	nb, no-NO-B	Step 1.3.6.1.4.1.42.2.27.9.4.110.1
Norwegian (Nynorsk nn)	nn, nn-NO	Step 1.3.6.1.4.1.42.2.27.9.4.109.1
Norwegian (Nynorsk no-NO-NY)	no-NO-NY	Step 1.3.6.1.4.1.42.2.27.9.4.108.1
Oromo	om	Step 1.3.6.1.4.1.42.2.27.9.4.111.1
Oromo (Ethiopia)	om-ET	Step 1.3.6.1.4.1.42.2.27.9.4.112.1
Oromo (Kenya)	om-KE	Step 1.3.6.1.4.1.42.2.27.9.4.113.1
Persian	fa	Step 1.3.6.1.4.1.42.2.27.9.4.71.1
Persian (India)	fa-IN	Step 1.3.6.1.4.1.42.2.27.9.4.72.1

Language by locale	Locale abbreviation	OID
Persian (Iran)	fa-IR	Step 1.3.6.1.4.1.42.2.27.9.4.73.1
Polish, Polish (Poland)	pl, pl-PL	Step 1.3.6.1.4.1.42.2.27.9.4.114.1
Portuguese, Portuguese (Portugal)	pt, pt-PT	Step 1.3.6.1.4.1.42.2.27.9.4.115.1
Portuguese (Brazil)	pt-BR	Step 1.3.6.1.4.1.42.2.27.9.4.116.1
Romanian, Romanian (Romania)	ro, ro-RO	Step 1.3.6.1.4.1.42.2.27.9.4.117.1
Russian, Russian (Russia)	ru, ru-RU	Step 1.3.6.1.4.1.42.2.27.9.4.118.1
Russian (Ukraine)	ru-UA	Step 1.3.6.1.4.1.42.2.27.9.4.119.1
Serbian, Serbian (Serbia)	sr, sr-RS	Step 1.3.6.1.4.1.42.2.27.9.4.128.1
Serbo-Croatian	sh	Step 1.3.6.1.4.1.42.2.27.9.4.120.1
Slovak, Slovak (Slovakia)	sk, sk-SK	Step 1.3.6.1.4.1.42.2.27.9.4.121.1
Slovenian, Slovenian (Slovenia)	sl, sl-SI	Step 1.3.6.1.4.1.42.2.27.9.4.122.1
Somali, Somali (Somalia)	so, so-SO	Step 1.3.6.1.4.1.42.2.27.9.4.123.1
Somali (Djibouti)	so-DJ	Step 1.3.6.1.4.1.42.2.27.9.4.124.1
Somali (Ethiopia)	so-ET	Step 1.3.6.1.4.1.42.2.27.9.4.125.1
Somali (Kenya)	so-KE	Step 1.3.6.1.4.1.42.2.27.9.4.126.1
Spanish, Spanish (Spain)	es, es-ES	Step 1.3.6.1.4.1.42.2.27.9.4.49.1
Spanish (Argentina)	es-AR	Step 1.3.6.1.4.1.42.2.27.9.4.50.1
Spanish (Bolivia)	es-BO	Step 1.3.6.1.4.1.42.2.27.9.4.51.1
Spanish (Chile)	es-CL	Step 1.3.6.1.4.1.42.2.27.9.4.52.1
Spanish (Colombia)	es-CO	Step 1.3.6.1.4.1.42.2.27.9.4.53.1
Spanish (Costa Rica)	es-CR	Step 1.3.6.1.4.1.42.2.27.9.4.54.1
Spanish (Dominican Republic)	es-DO	Step 1.3.6.1.4.1.42.2.27.9.4.55.1
Spanish (Ecuador)	es-EC	Step 1.3.6.1.4.1.42.2.27.9.4.56.1
Spanish (El Salvador)	es-SV	Step 1.3.6.1.4.1.42.2.27.9.4.65.1

Language by locale	Locale abbreviation	OID
Spanish (Guatemala)	es-GT	Step 1.3.6.1.4.1.42.2.27.9.4.57.1
Spanish (Honduras)	es-HN	Step 1.3.6.1.4.1.42.2.27.9.4.58.1
Spanish (Mexico)	es-MX	Step 1.3.6.1.4.1.42.2.27.9.4.59.1
Spanish (Nicaragua)	es-NI	Step 1.3.6.1.4.1.42.2.27.9.4.60.1
Spanish (Panama)	es-PA	Step 1.3.6.1.4.1.42.2.27.9.4.61.1
Spanish (Paraguay)	es-PY	Step 1.3.6.1.4.1.42.2.27.9.4.64.1
Spanish (Peru)	es-PE	Step 1.3.6.1.4.1.42.2.27.9.4.62.1
Spanish (Puerto Rico)	es-PR	Step 1.3.6.1.4.1.42.2.27.9.4.63.1
Spanish (United States)	es-US	Step 1.3.6.1.4.1.42.2.27.9.4.66.1
Spanish (Uruguay)	es-UY	Step 1.3.6.1.4.1.42.2.27.9.4.67.1
Spanish (Venezuela)	es-VE	Step 1.3.6.1.4.1.42.2.27.9.4.68.1
Swahili	sw	Step 1.3.6.1.4.1.42.2.27.9.4.131.1
Swahili (Kenya)	sw-KE	Step 1.3.6.1.4.1.42.2.27.9.4.132.1
Swahili (Tanzania)	sw-TZ	Step 1.3.6.1.4.1.42.2.27.9.4.133.1
Swedish, Swedish (Sweden)	sv, sv-SE	Step 1.3.6.1.4.1.42.2.27.9.4.129.1
Swedish (Finland)	sv-FI	Step 1.3.6.1.4.1.42.2.27.9.4.130.1
Tamil	ta	Step 1.3.6.1.4.1.42.2.27.9.4.134.1
Telugu	te	Step 1.3.6.1.4.1.42.2.27.9.4.135.1
Thai, Thai (Thailand)	th, th-TH	Step 1.3.6.1.4.1.42.2.27.9.4.136.1
Tigrinya	ti	Step 1.3.6.1.4.1.42.2.27.9.4.137.1
Tigrinya (Eritrea)	ti-ER	Step 1.3.6.1.4.1.42.2.27.9.4.138.1
Tigrinya (Ethiopia)	ti-ET	Step 1.3.6.1.4.1.42.2.27.9.4.139.1
Turkish, Turkish (Türkiye)	tr, tr-TR	Step 1.3.6.1.4.1.42.2.27.9.4.140.1
Ukrainian, Ukrainian (Ukraine)	uk, uk-UA	Step 1.3.6.1.4.1.42.2.27.9.4.141.1

Language by locale	Locale abbreviation	OID
Vietnamese, Vietnamese (Vietnam)	vi, vi-VN	Step 1.3.6.1.4.1.42.2.27.9.4.142.1

Working with the matching entry count control

You can use the `ldapsearch` command with the `--matchingEntryCountControl` option to determine the count of entries that match a search filter.

You must grant users access to this control by invoking 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\";)"
```

You can use an `examineCount` control for searches that are at least partially indexed to determine whether to return an examined count, an unexamined count, or an upper bound count. The following factors determine what the search returns:

- A search is fully indexed if indexes can be used to identify the entry IDs for all entries that match the filter without ambiguity. You can also use indexes to make sure that all of those candidates are within the scope of the search.
- A search is partially indexed if you can use indexes to identify the entry IDs for all entries that match the search criteria.

  > **Note**
  >
  > The candidate list could also include entries that either don't match the filter or are outside the scope of the search.

- A search is unindexed if it's 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 the search would return them to the client.

  An examined count might be less than an unexamined count if the set of matching entries includes those that the search would remove, including special entries, such as LDAP subentries, replication conflict entries, or soft-deleted entries, and entries excluded through access control evaluation.
- 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 doesn't 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 server processes the unindexed search, which can be very expensive. Instead of getting the matching entries back, the server returns the examined count. If `allowUnindexed` is false, the server returns an unknown count. If `allowUnindexed` is true, the requester needs to have the `unindexed-search` privilege to get the exact count.

Example

The following example demonstrates an `ldapsearch` using the `--matchingEntryCountControl` option:

```
$./ldapsearch \
  --bindDN "cn=directory manager" \
  --bindPassword password \
  --baseDN dc=example,dc=com \
  --matchingEntryCountControl examineCount=100:alwaysExamine:allowUnindexed:debug \
  "(objectclass=inetOrgPerson)"
```

Working with the entry counter plugin

You can use the entry counter plugin to count and report the number of entries in the server that match your specified criteria.

How the entry counter plugin works

The plugin periodically performs background searches to find entries matching your configured search criteria, which can include base DNs, filters, and named subcategory filters. The plugin reports the number of matching entries and other related information in monitor entries that you can retrieve using the admin console or an LDAP client.

The entry counter plugin minimizes the need for long, expensive searches. The plugin updates its search results after each polling duration, which is configured using `time-between-searches`.

> **Tip**
>
> Use the entry counter plugin to create a search for active users so that you can monitor your need for user licenses in real time.

How to set up the entry counter plugin

By default, the entry counter plugin isn't enabled. You need to do the following:

1. Create and enable the plugin.
2. Configure a set of criteria for the plugin.

3. Consume the output of the plugin.

> **Note**
>
> You can only enable and configure a single instance of the plugin, but you can create multiple named sets of criteria.

Creating and enabling the plugin

When creating the entry counter plugin, do the following:

- Give the plugin a name, which you reference when creating criteria.
- Specify a plugin type of `entry-counter`.
- Enable the plugin.
- (Optional) Specify the length of time between searches using `time-between-searches`.

> **Note**
>
> The `time-between-searches` parameter represents the length of time between the end of one search and the beginning of the next. The default value is 1 hour.
> Before you set a different value, consider how many entries a search involves and how long it could take. Set a value that accounts for both the length of the search and your need for the latest information.

Here's an example of how to create and enable the entry counter plugin:

```
$ bin/dsconfig create-plugin \
    --plugin-name "Entry Counter" \
    --type entry-counter \
    --set enabled:true \
    --set time-between-searches:100ms
```

Configuring a set of entry counter criteria

You can configure one or more sets of entry counter criteria. Each set can specify the following:

- Which entries to count (required)
- Warning and error count thresholds (optional)
- Whether to report entry sizes on disk (optional)

Use the following properties to configure a set of entry counter plugin criteria:

Property	Required or optional	Description
`base-dn`	Optional	A multivalued property that specifies one or more base DNs to use when searching for matching entries. If you don't specify a base DN, the server uses the base DNs for all public backends defined in the server.

Property	Required or optional	Description
`filter`	**Required**	**A single-valued property that specifies the filter to use when searching for matching entries.** **For example, to identify user entries, you can set a filter like `"(|(objectClass=person)(objectClass=ubidPerson))"`.** **Note** Different environments could use different object classes for user entries.

`named-sub-category-filter`	**Optional**	A multivalued property that specifies filters that subcategorize matching entries using additional criteria. Use the form `name:filter` to set values for this property. For example, if you set the `filter` property to find user entries in the server, you can use the `named-sub-category-filter` property to further categorize those entries into active users, inactive users, or users who have never authenticated. In the following samples, these categories are determined based on the last time a user successfully authenticated, where the value `31536000` represents the number of seconds in 1 year:
		`active-users:(ds-pwp-state-json:jsonObjectFilterExtensibleMatch:={ "filterType":"lessThan", "field":"seconds-since-last-login", "value":31536000, "allowEquals":true })`
		`inactive-users:(ds-pwp-state-json:jsonObjectFilterExtensibleMatch:={ "filterType":"greaterThan", "field":"seconds-since-last-login", "value":31536000, "allowEquals":false })`
		`users-who-have-never-authenticated:(ds-pwp-state-json:jsonObjectFilterExtensibleMatch:={ "filterType":"negate", "negateFilter":{ "filterType":"containsField", "field":"last-login-time" } })`

`warning-threshold-minimum-count`	**Optional**	**A single-valued property that specifies a minimum threshold of matching entries to raise a warning alarm.** **The number of matching entries is based on the combination of base DN and filter criteria.** **Note** The server doesn't raise a warning alarm if the `error-threshold-minimum-count` threshold is also triggered.
`error-threshold-minimum-count`	**Optional**	**A single-valued property that specifies a minimum threshold of matching entries to raise an error alarm.** **The number of matching entries is based on the combination of base DN and filter criteria.**

Property	Required or optional	Description
`track-matching-entry-size`	Optional	A Boolean property that specifies whether the server tracks the amount of space used to store information about the entries in the backend database. Set the value to `true` to enable this property. **Important** If this property is enabled, the entry counter processing gets more expensive because the server performs additional operations to calculate the size of each entry. The matching entry size returned only represents the size of the encoded entry itself and doesn't include any associated index records.

When configuring a set of criteria for the entry counter plugin, do the following:

- Reference the name you gave the plugin using `plugin-name`.
- Give the set of criteria a name using `criteria-name`.
- Set the search criteria.
- If needed, configure count thresholds and entry size reporting.

Important

The plugin treats each set of criteria separately and returns a monitor entry for each set.

Here's an example of how to configure a set of entry counter plugin criteria:

```
$ bin/dsconfig create-entry-counter-plugin-criteria \
    --plugin-name "Entry Counter" \
    --criteria-name Users \
    --set base-dn:dc=example,dc=com \
    --set filter:(objectClass=person)
```

Here's an example of how to delete a set of entry counter plugin criteria:

```
$ bin/dsconfig delete-entry-counter-plugin-criteria \
    --plugin-name "Entry Counter" \
    --criteria-name Users
```

Consuming the output of the plugin

Monitor entries generated from the plugin criteria have a structural object class of `ds-entry-counter-plugin-monitor-entry` and can include the following attributes with values specific to their associated criteria.

> **Tip**
>
> Learn more about adding monitor entries to the admin console.

Attribute	Present in the entry?	Description
`criteria-name`	Always	The name of the set of criteria specified for the monitor entry.
`results-available`	Always	Indicates whether the server has completed at least one internal search to identify the number of matching entries. Use this attribute to determine whether entry count data will be available for the criteria. **Note** At server startup, after the plugin is enabled, or after new criteria is defined, the server won't have completed any entry counter searches, but it will still create a monitor entry for that criteria.
`results-last-updated-time`	Only with results	Indicates the time that the matching entry count results were last updated.
`base-dn`	Always	The included base DNs (or the default set, if none were configured).
`filter`	Always	The filter from the search criteria.
`matching-entry-count`	Only with results	The number of entries that match the criteria.
`sub-category-filter-json`	Always (if defined)	A multivalued JSON attribute with information about any subcategory filters defined in the criteria. Each filter is represented by a separate value that contains the following JSON fields: **`name`** The name of the subcategory filter (always included). **`filter`** The filter used to subcategorize matching entries (always included). **`count`** The number of entries that match the filter (only with results).
`warning-threshold-minimum-matching-entry-count`	Always (if defined)	The minimum matching entry count threshold for the server to raise a warning alarm.

Attribute	Present in the entry?	Description
`warning-threshold-reached`	Only with results (if defined)	Indicates whether the number of entries that match the criteria is greater than or equal to the warning threshold.
`percent-of-warning-threshold-reached`	Only with results (if defined)	A percentage that indicates how close the number of matching entries is to reaching the warning threshold. The value can be greater than 100% if the number of matching entries exceeds the warning threshold.
`error-threshold-minimum-matching-entry-count`	Always (if defined)	The minimum matching entry count threshold for the server to raise an error alarm.
`error-threshold-reached`	Only with results (if defined)	Indicates whether the number of entries that match the criteria is greater than or equal to the error threshold.
`percent-of-error-threshold-reached`	Only with results (if defined)	A percentage that indicates how close the number of matching entries is to reaching the error threshold. The value can be greater than 100% if the number of matching entries exceeds the error threshold.
`track-matching-entry-size`	Always	Indicates whether the `track-matching-entry-size` property is enabled.
`total-matching-entry-size-bytes`	Only with results (if enabled)	The total size, in bytes, required to store all matching entries in the database.
`average-matching-entry-size-bytes`	Only with results (if enabled)	The average size, in bytes, required to store each matching entry in the database.
`minimum-matching-entry-size-bytes`	Only with results that include at least one matching entry (if enabled)	The size, in bytes, required to store the smallest matching entry in the database.
`maximum-matching-entry-size-bytes`	Only with results that include at least one matching entry (if enabled)	The size, in bytes, required to store the largest matching entry in the database.

Adding entries

Depending on the number of entries that you want to add to your PingDirectory server, you can use the `ldapmodify` tool for small additions.

The `ldapmodify` tool provides two methods for adding a single entry:

- Using an LDIF file
- Using 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 enables a client to include attributes that are not normally allowed from external sources, such as the `userPassword` attribute, which 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 relative distinguished name (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.
> 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.
> You can also use the uniqueness request control 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

Use the `ldapmodify` tool to add an entry from an LDIF file.

Steps

1. Open a text editor and create an entry that conforms with your schema.

   > **Note**
   >
   > The PingDirectory 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.

   Example:

   The following example adds the entry in the file and saves the file as `add-user.ldif`. For the `userPassword` attribute, enter the cleartext password.

```
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. To add the entry specified in the LDIF file, run the `ldapmodify` tool.

   Example:

   ```
   $ bin/ldapmodify --defaultAdd --filename add-user.ldif
   ```

   Result:

   A confirmation message of the new addition appears. If the command is successful, you'll see generated success messages with the "#" symbol.

   ```
   # 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 you can use 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 `add` and `modify` in one file.

Steps

1. Open a text editor and create an entry that conforms with your schema.

   Example:

This example uses `changetype: add` to add the following entry in the file and saves the file as `add-user2.ldif`.

```
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. To add the entry specified in the LDIF file, run the `ldapmodify` tool.

   Example:

   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
   ```

   Result:

   A confirmation message displays confirming the addition.

Adding multiple entries in a single file

About this task

Add multiple entries in your LDIF file by separating each distinguished name (DN) and its entry with a blank line from the next entry.

Steps

1. Open a text editor and create some entries that conform to your schema.

   Example:

   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. To add the entries specified in the LDIF file, run the `ldapmodify` tool.

   Example:

   This example uses the short form arguments for the `ldapmodify` tool.

   ```
   $ bin/ldapmodify -h server.example.com -p 389 \
     -D "cn=admin,dc=example,dc=com" -w password -a -f add-user3.ldif
   ```

   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 are treated as add operations, and `-f` specifies the path to the input file. If the operation is successful, you will see commented messages (those beginning with "#") for each addition.

   Result:

   ```
   # 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.

> **Note**
>
> Ensure that there are no child entries below the entry because that can create an orphaned entry.

Back up your system before removing any entries.

Deleting an entry using ldapdelete

Steps

1. To delete an entry, run the `ldapdelete` command.

   Example:

   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

About this task

To generate a file of distinguished names (DNs) to delete from the PingDirectory server:

Steps

1. To search for all entries in a branch and return the DNs of the subentries, run the following command.

   Example:

   For this example, the search is for all entries in the `ou=Accounting` branch.

   ```
   $ 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.

   Example:

   The following command uses the `--continueOnError` option, which continues deleting through the whole list even if an error is encountered for a DN entry.

   ```
   $ bin/ldapdelete --filename /usr/local/entry_dns.txt --continueOnError
   ```

Deleting entries using ldapmodify

About this task

You can use the LDIF changetype directive to delete an entry from the PingDirectory server using the `ldapmodify` tool.

> **Note**
>
> You can only delete leaf entries.

Steps

1. Delete an entry using the `ldapmodify` tool.
    1. From the command line run the `ldapmodify` tool with the `changetype:delete` option.
    2. Enter the distinguished name (DN) and press `Enter` to go to the next line.
    3. Enter the `changetype` directive.
    4. Press `Ctrl+D` twice to enter the end-of-file (EOF) sequence (UNIX) or `Ctrl+Z` (Windows).

        Example:

        ```
        $ 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. To locate a specific entry, run the `ldapsearch` tool.

    Example:

    ```
    $ 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. To change attributes from the command line, run the `ldapmodify` command.
    1. Specify the modification using the `changetype:modify` directive and then specify which attributes are to be changed using the `replace` directive.

        Example:

        In this example, we change the telephone number of a specific user entry.

```
$ 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
```

3. Process the request:

   Choose from:

   - For Unix EOF escape sequence, enter `ctrl+d` twice.
   - For Windows, enter `ctrl+z`.

Modifying multiple attributes in an entry from the command line

Steps

1. To locate a specific entry, run the `ldapsearch` tool.

   Example:

   ```
   $ 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. To change attributes from the command line, use the `ldapmodify` command.

   1. Specify the modification using the `changetype:modify` subcommand.
   2. Specify the attributes to change using the `add` and `replace` subcommand.

      Example:

      In this example, we add the `postOfficeBox` attribute and 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 ("-").

      ```
      $ 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
      ```

3. Process the request.

Choose from:

- For Unix, enter `ctrl+d` twice.

  This is the Unix EOF escape sequence.

- For Windows, enter `ctrl+z`.

Adding an attribute from the command line

Steps

1. Run the `ldapmodify` tool from the command line interface.
2. Specify the modification using the `changetype:modify` subcommand.
3. Specify which attributes to add using the `add` option.

   Example:

   In this example, we add another value for the `cn` attribute, which is multi-valued.

   ```
   $ 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
   ```

4. To process the request, enter `ctrl+d` twice.

   This is the UNIX EOF escape sequence.

   > **Note**
   >
   > An error might 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.
   > Remove any trailing spaces after the attribute value.

Adding an attribute using the language subtype

About this task

The PingDirectory 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

1. Run the `ldapmodify` tool from the command line interface.
2. Specify the modification using `changetype:modify`.
3. Specify which attributes to add using the `add` option.

   Example:

In this example, we add another value for the `lang` attribute.

```
$ 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
```

4. To process the request, enter `ctrl+d` twice.

   This is the UNIX EOF escape sequence.

Adding an attribute using the binary subtype

About this task

The PingDirectory server provides support for attributes using binary subtypes, which are typically used for certificates or JPEG images that you can store in an entry. The operation must specifically match the subtype for successful operation. You must use the version directive with a value of "1" for binary subtypes. Typical binary attribute types are `userCertificate` and `jpegPhoto`.

Steps

1. To add an attribute with a binary subtype, run the `ldapmodify` tool from the command line interface.
2. Specify the modification using `changetype:modify`.
3. Specify which attributes to add using the `add` option.

   Example:

   The attribute in this example points to the filepath 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

- To delete an attribute, run the `ldapmodify` tool with the LDIF `delete` subcommand.

  Example:

```
$ 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

Use the LDIF `delete` subcommand to delete a specific attribute value from an attribute.

Steps

1. Run the `ldapmodify` tool from the command line.
2. Specify the modification using `changetype:modify`.
3. Specify the attribute pair that you want to delete using `delete`.

   Example:

   The following sample assumes you have multiple values of `cn` in an entry, such as `cn: Sally Tree`, `cn: Sally Tea Tree`. This sample keeps `cn:Sally Tree` and deletes 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

Rename an entry by changing the relative distinguished name (RDN) of an entry.

About this task

> **Note**
>
> You cannot rename an RDN if it has children entries because this violates the LDAP protocol.

Steps

- Run the `ldapmodify` tool to rename an entry.
  - Run the `changetype`, `newrdn`, and `deleteoldrdn` directives.

    Example:

    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 PingDirectory server

Run the `ldapmodify` tool to move an entry from one base distinguished name (DN) to another base DN.

Before you begin

- 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)`.

About this task

For more information on access control instructions, see Overview of access control.

Steps

1. To move an entry from one branch to another, run the `ldapmodify` tool.

   Example:

   In this example, an entry moves from the `ou=contractors` 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
   ```

2. Specify the modification using `changetype:moddn`.

Moving an entry from one machine to another

About this task

The PingDirectory server provides the `move-subtree` tool to move a subtree or one entry on one machine to another.

The `move-subtree` tool moves a subtree or multiple entries from one machine to another. The tool does not copy the entries. After the entries are moved, they are no longer present on the source server.

> **Note**
>
> The subtree or entry must exist on the source server and cannot 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.

Steps

- To move an entry, such as `uid=test.user,ou=People,dc=example,dc=com`, from the source host to the target host, run the `move-subtree` tool.

  Example:

```
$ 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 distinguished name (DN) file from one machine to another. Lines that begin with the octothorpe character (#) and empty lines will be ignored. You can prefix entry DNs with `dn:`, but long DNs can'tt be wrapped across multiple lines.

Steps

1. In a text editor, create a new 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.

   Example:

   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. To specify the file of DNs to move from one machine to another, run the `move-subtree` tool with the `--entryDNFile` option.

   Example:

```
$ 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
```

> **Note**
>
> If an error occurs with one of the DNs in the file, the output message shows the error. The `move-subtree` tool keeps 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, such as 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, such as if 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 applies the changes and reaches the end of the LDIF file, it automatically displays the update statistics described in the following table.

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

Processing Statistic	Description
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.

   Example:

   The third change in this example generates 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. To apply the changes in the LDIF file to a target server, run `parallel-update` with the `--ldifFile` and `--rejectFile` options.

   > **Note**
   >
   > The `--ldifFile` and `--rejectFile` options are required.

   Example:

In this example, there are ten concurrent threads. The optimal number of threads depends on your underlying system.

```
$ bin/parallel-update --hostname 127.0.0.1 \
  --ldifFile changes.ldif --rejectFile reject.ldif --numThreads 10
```

Result:

```
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 files for any failed updates.

   Result:

   The following sample elaborates on the rejected file.

```
# ResultCode=53, Diagnostic Message=Pre-encoded passwords are not allowed for
# the password attribute userPassword
dn: uid=user.101,ou=People,dc=example,dc=com
changetype: add
objectClass: person
objectClass: inetOrgPerson
objectClass: organizationalPerson
objectClass: top
postalAddress: Ziggy Zad$15172 Monroe Street$Salt Lake City, MI 49843
postalCode: 49843
description: This is the description for Ziggy Zad.
uid: user.101
userPassword: {SSHA}IK57iPozIQybmIJMMdRQOpIRudIDn2RcF6bDMg==
```

Working with the watch-entry tool

The PingDirectory server provides a `watch-entry` tool that demonstrates replication or synchronization latency by watching an LDAP entry for changes.

About this task

You can directly modify attributes with the `watch-entry` tool.

> **Note**
>
> If the LDAP entry changes, the background of modified attributes will briefly be red.

Steps

- Run the `watch-entry` tool.

Example:

This example connects to `server.example.com` as `uid=admin,dc=example,dc=com` and watches 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 together at commit time.

Applications developed to perform batched transactions should include as few operations in the transaction as possible. The changes aren't actually processed until the commit request is received. Therefore, the client can't know whether the changes are successful until commit time. If any of the operations fail, then the entire set of operations fails.

Batched transactions are write operations that are processed as a single atomic unit when the commit request is received. The write operations include the following:

- `add`
- `delete`
- `modify`
- `modify DN`
- `password modify`

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's not currently possible to process a transaction that requires changes to be processed across multiple servers or multiple PingDirectory server backends.

Directory servers can limit the set of controls that are available for use in requests that are part of a transaction. RFC 5805 section 4 indicates that you can use the following controls in conjunction with the transaction specification request control:

- Assertion request control
- manageDsaIT request control
- Pre-read request control
- Post-read request control

The proxied authorization v1 and v2 controls can't 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

About this task

You can run the `ldapmodify` tool's `--useTransaction` option to process multiple operations as part of a single batched transaction.

Steps

1. Create a batch text file with the changes that you want to apply as a single atomic unit.

   Example:

   ```
   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. To run the batched transaction, run the `ldapmodify` tool with the `--useTransaction` and `--filename` options.

   Example:

   ```
   $ bin/ldapmodify --useTransaction --filename test.ldif
   ```

   Result:

   ```
   #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

PingDirectory server supports virtual attributes.

Virtual attributes are abstract, dynamically generated attributes that are invoked through an LDAP operation, such as `ldapsearch`, but are not stored in the PingDirectory server backend.

Most virtual attributes are operational attributes, providing processing-related information that the server requires. The PingDirectory server virtual attribute subsystem allows you to create user-defined virtual attributes to suit your requirements.

For example, you can mirror the values of other attributes (in the same or different entries), and values can be constructed from other attributes in the same entry. Items such as password policies and privileges can be dynamically generated, and you can create custom virtual attribute providers with the Server SDK using a wide variety of logic.

Viewing the list of default virtual attributes

Use the `dsconfig` tool to view the list of PingDirectory server virtual attributes.

Some virtual attributes are enabled by default and are useful for most applications.

The default set of virtual attributes is 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 PingDirectory 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.

Virtual Attributes	Description
`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`.
`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** 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 Attributes	Description
`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 because 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

- To list the virtual attributes, use `dsconfig`.

```
$ bin/dsconfig list-virtual-attributes
```

Viewing virtual attribute properties

View the basic properties of each virtual attribute using the `dsconfig` tool.

About this task

For a complete list of properties, see General Consent Service configuration. The following table shows some basic properties.

Property	Description
`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.

Property	Description
`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.

Steps

- To view the properties of a virtual attribute, use `dsconfig`.

  Example:

  ```
  $ 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 from the `Standard object menu`.

Enabling a virtual attribute using `dsconfig` interactive mode

Steps

1. To enable a virtual attribute, use `dsconfig`.
   1. Specify the connection port, bind DN, password, and host information.
   2. Enter the LDAP connection parameter for your PingDirectory server:
      - For LDAP, enter `1`.
      - For SSL, enter `2`.
      - For StartTLS, enter `3`.

      Example:

      ```
      bin/dsconfig
      ```

2. On the PingDirectory server main menu, enter `o` to change the object menu, and then enter the number corresponding to `Standard`.
3. On the PingDirectory server main menu, enter the number corresponding to virtual attributes.
4. On the `Virtual Attribute management` menu, enter 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.

   For this example, enter the number corresponding to the `numSubordinates` virtual attribute.
6. On the `numSubordinates Virtual Attribute Properties` menu, enter the number to enable the virtual attribute.
7. On the `Enabled Property` menu for the `numSubordinates` virtual attribute, enter the number to change the value to `TRUE`.
8. On the `numSubordinates Virtual Attribute Properties` menu, enter `f` to apply the changes.
9. Verify that the virtual attribute is enabled.

   Example:

   > **Note**
   >
   > This example assumes you have configured the group entries.

   ```
   $ bin/ldapsearch --baseDN dc=example,dc=com "(ou=People)" numSubordinates
   ```

   Result:

   The system returns the following.

   ```
   dn: ou=People,dc=example,dc=com
   numSubordinates: 1000
   ```

Enabling a virtual attribute using `dsconfig` non-interactive mode

Steps

- To enable a virtual attribute, such as `numSubordinates`, use `dsconfig`.

  Example:

  ```
  $ 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, or you will see an error message when you configure it.

You can define your virtual attributes using the `dsconfig` tool from the `Standard object menu`. Only the value property is specific to the user-defined virtual attribute. All of the other properties are common across all kinds of virtual attributes, which include the following:

Property	Description
`enabled`	Indicates whether the virtual attribute should be used.
`attribute-type`	The attribute type for the virtual attribute that is generated.

`base-dn`
`group-dn`
`filter`
	Can be used to select which entries are eligible to contain the virtual attribute.

`client-connection-policy`	Can be used to select which entries are eligible to contain the virtual attribute.
`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 values, only the virtual values, 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 of the server if multiple virtual attributes can 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. To configure the user-defined virtual attribute:
    1. Run `dsconfig`.
    2. Specify the connection port, bind DN, password, and host information.

3. Type the LDAP connection parameter for your PingDirectory server:
    - For LDAP, enter `1`.
    - For SSL, enter `2`.
    - For StartTLS, enter `3`.
2. To change the object menu, in the PingDirectory server main menu, type `o`, and then type the number to select `Standard`.
3. In the PingDirectory server main menu, type the number corresponding to virtual attributes.
4. To create a new virtual attribute, in the `Virtual Attribute management` menu, type the number.
5. Use an existing virtual attribute as a template for your new attribute, or create a new attribute from scratch.

   In this example, type `n` to create a new Virtual Attribute from scratch.
6. In 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. Enter a name for the new virtual attribute.

   In this example, enter `Employee Password Policy Assignment`.
8. In the `Enabled Property` menu, enter the number to set the property to `TRUE` (enable).
9. In 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. In the `User Defined Virtual Attributes` menu, enter a description for the virtual attribute.

    Though optional, this step is useful if you plan to create multiple virtual attributes. Enter the option to change the value, and then type a description of the virtual attribute. In this example, type `Virtual attribute that assigns the Employee Password Policy to all entries that match (employeeType=employee)`.
12. In the `User Defined Virtual Attribute` menu, type the number corresponding to the filter.
13. In 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. In the `User Defined Virtual Attribute` menu, type `f` to finish creating the virtual attribute.
15. Verify that the attribute was created successfully.
    1. Add the `employeeType=employee` attribute to an entry, such as `uid=user.0`, using **`ldapmodify`**.

2. Add the `employeeType=contractor` attribute to another entry, such as `uid=user.1`.

16. To search for the user with the `employeeType=employee` attribute, such as `uid=user.0`, use `ldapsearch`.

Example:

```
$ bin/ldapsearch --baseDN dc=example,dc=com "(uid=user.0)" \
  ds-pwp-password-policy-dn
```

Result:

The `ds-pwp-password-policy-dn` attribute has the assigned password policy as its value.

```
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)`.

Example:

```
$ bin/ldapsearch --baseDN dc=example,dc=com "(uid=user.1)" \
  ds-pwp-password-policy-dn
```

Result:

The `ds-pwp-password-policy-dn` attribute is not present in the entry because the entry has the attribute `employeeType=contractor`.

```
dn: uid=user.1,ou=People,dc=example,dc=com
```

Creating a user-defined virtual attribute using `dsconfig` in non-interactive mode

About this task

You can create a virtual attribute in non-interactive command-line mode using `dsconfig`.

Steps

- To set up the Employee Password Policy Assignment virtual attribute introduced in the previous section, run the following command.

Example:

```
$ 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

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, consider a directory information tree (DIT) where users have a `manager` attribute 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.

```
$ 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

Create a mirror virtual attribute using `dsconfig` in non-interactive command-line mode.

About this task

The following example sets up the `managerName` virtual attribute introduced in the previous section:

Steps

1. Update the schema to define the `managerName` attribute.

   You can optionally add the attribute to an object class.

   1. In a text editor, create a file with the following schema definition for the attribute.

```
dn: cn=schema
objectClass: top
objectClass: ldapSubentry
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 'PingDirectory Server Example' )
```

2. Save the file as `98-myschema.ldif` in the `<server-root>/config/schema` folder.

2. Restart the PingDirectory server.

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

3. To create the virtual attribute, use `dsconfig`.

```
$ 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. To verify the mirror virtual attribute, search for an entry.

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

Result:

Your search results in the following.

```
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 with the `dsconfig` tool.

Make sure 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 from the `Standard object menu`.

Editing a virtual attribute using dsconfig in non-interactive mode

Steps

- To change a property value, use `dsconfig`.

  Example:

  ```
  $ bin/dsconfig set-virtual-attribute-prop --name dept-number \
    --set "value:111"
  ```

Deleting a virtual attribute

Delete virtual attributes using the `dsconfig` tool.

About this task

If you are using `dsconfig` in interactive command-line mode, you can access the virtual attribute menu from the `Standard object menu`.

Steps

- To delete the `dept-number` virtual attribute, use `dsconfig`.

  ```
  $ bin/dsconfig delete-virtual-attribute --name dept-number
  ```

Virtual attribute limitations

This section describes the limitations of virtual attributes, such as performance, indexing, and consistency limitations.

Performance limitations

The performance impact of virtual attribute generation is limited by computing these values only if they are returned to the client. However, performance can still be impacted because of other factors. Virtual attributes might be included or excluded based on a range of criteria and the evaluation of this criteria (for example, based on group membership) can impact performance.

Indexing limitations

You cannot index the values of a virtual attribute because the values can change from one invocation to the next.

Some virtual attributes implement support for an index-like functionality by offering logic that can quickly identify entries with 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 retrieves the entry with the specified distinguished name (DN).
- The `isMemberOf` virtual attribute provider uses the server's group manager to identify the members of a targeted group.

> **Note**
>
> This is not an option for all types of virtual attributes.

Index-like functionality is not available for the following virtual attribute types:

- User-defined virtual attributes that have a static value
- Constructed virtual attributes whose values are constructed from multiple other attributes in the same entry
- Mirror virtual attributes whose values come from other entries
- Any custom virtual attribute types implemented using the Server SDK

If you want to have a mix of standard and virtual values for the same attribute type, whether in the same entry or in different entries, there is not a good solution. If you define an index for that attribute type, it only includes entries with standard values and entries with virtual values are overlooked. If there is no index but the virtual attribute provider supports some search functionality, only entries with virtual values are matched and entries with only standard values are excluded.

Unexpected behavior for write operations

A significant issue with virtual attributes is that they can result in unexpected behaviors when targeted by write operations.

These unexpected behaviors include the following:

- Attempting to delete a virtual value fails 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 causes 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 causes 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 yields a success result, but the operation has 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 yields a success result, but only the real values are replaced and the virtual values 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 only applies to operational attribute types. This is not an acceptable limitation in many cases. Access control restrictions could work for many clients, but do not have any effect for requesters with the `bypass-acl` privilege.

Working with composed attributes

This section provides information on composed attributes, which generate attribute values to complement virtual attributes.

PingDirectory supports virtual attributes, which you can use to dynamically generate values in entries for a wide range of use cases, including values for key operational attributes. Composed attributes complement virtual attributes and are designed for scenarios that virtual attributes don't support.

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 entries can be created, updated, or removed when processing `modify` and `modifyDN` operations. The server might optionally permit or reject attempts to alter the generated values.

Composed values are based on a combination of static text and the contents of the entry in which the value is being generated. You cannot compose values using content from other entries in the server because the necessary content might not be available at the time the value is being generated. An example of this is if the referenced entry does not yet exist. It would also be difficult to ensure that the value 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 and 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 `modifyDN` operations and also operates during LDIF import.

The server provides an administrative task that can be used to iterate through all entries in a backend and generate values. This is useful when a new composed attribute plugin instance is created after the backend has already been populated and you do not want 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 limited to what the server can provide. You can implement your own plugins to generate your own data.

Composed attribute plugin configuration properties

The composed attribute plugin provides the following configuration properties.

Configuration properties

`plugin-type`

The plugin type values for which the server is registered.

This is a mandatory multi-valued property with a default that includes all of the `ldifimport`, `preoperationadd`, `preoperationmodify`, and `preoperationmodifydn` values. In general, you should not override this default set, but this might 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 distinguished name (DN) operations.

`attribute-type`

The name or OID of the attribute type for which values are generated.

This attribute type must be defined in the schema. This is a mandatory, single-valued property with no default. To have multiple composed attribute types, configure a separate instance of the plugin for each attribute type. To compose multiple values for the same attribute type from different sets of source attributes, use a single instance of the plugin with multiple value patterns.

`value-pattern`

The value pattern that specifies the logic that should be used when composing 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 is preserved. If this is configured with multiple values, then the behavior the plugin should exhibit is controlled by the `multiple-value-pattern-behavior` property.

> **Note**
>
> A value pattern is only used for an entry if that entry contains all of the attributes referenced in the value pattern. 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 is a single-valued property with a default value of `use-first-non-rejected-value-pattern-with-non-empty-values-but-might-reject`.

> **Note**
>
> The primary reason that a rejection might 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-might-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 is generated. If no value pattern yields a non-empty result but at least one is rejected, then the operation fails, or the entry is 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 is 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 is rejected.

If none of the value patterns is not rejected and yields a non-empty result, then no composed value is generated.

`use-all-non-rejected-value-patterns-with-non-empty-values-but-might-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 is used. If none of the value patterns yielded non-empty results but none were rejected, then no composed value is generated. If none of the value patterns yielded non-empty results and at least one was rejected, then the operation fails or the LDIF entry is 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 is used. If none of the value patterns yielded non-empty results, regardless of whether any of them was rejected, then no composed value is 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 fails or the LDIF entry is rejected. If none of the value patterns is rejected but none of them generates any values, then no composed values are 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**
>
> No more than one source attribute should 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 might fail.

This is 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 is 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 is used so that the entry is not 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 is 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 compose 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 is discarded.

`merge-existing-and-composed-values`

Indicates that the server should preserve any existing values for the target attribute and should compose any additional values.

> **Note**
>
> This option can 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 should preserve any existing values when performing an LDIF import.

> **Note**
>
> The server does 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 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 `modifyDN` operations that alter values for one of the attributes used in the value pattern.

This is 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 are 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 is a single-valued property with a default of `preserve-non-composed-values`. The available options include:

preserve-non-composed-values

Indicates that the server attempts to remove any values for the target attribute that would have been generated for that entry by the associated value patterns but preserves any other values that might exist for the target attribute.

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 are preserved to avoid violating the schema.

preserve-all-values

Indicates that the server preserves 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 attempts to remove all values for the target attribute from the entry, regardless of whether they would have been composed but falls 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 attempts to remove all values for the target attribute from the entry, regardless of whether they would have been composed but falls 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 `modifyDN` operation.

This is a single-valued property with a default of `always-allow`. The available options include:

always-allow

Indicates that the server always permits 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 only permits clients to alter values for the target attribute that were not generated by the plugin.

Attempts to alter composed values are rejected.

never-allow

Indicates that the server should reject any attempt to alter the target attribute regardless of the values it might have.

`include-base-dn`

A base DN that might be used to restrict the set of entries for which values are composed.

The following section provides a detailed description of the include and exclude criteria configuration options.

`exclude-base-dn`

A base DN that might be used to restrict the set of entries for which values are composed.

The following section provides a detailed description of the include and exclude criteria configuration options.

`include-filter`

A search filter that might be used to restrict the set of entries for which values are composed.

The following section provides a detailed description of the include and exclude criteria configuration options.

`exclude-filter`

A search filter that might be used to restrict the set of entries for which values are composed.

The following section provides a detailed description of the include and exclude criteria configuration options.

`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 is 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 are preserved and no additional composed values are added.

If the entry did not contain the target attribute, then no composed values are 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 are preserved and no additional composed values are added.

If the entry did not contain the target attribute, then composed values are 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 are preserved and any additional values that would be composed for that entry is also added.

If the entry did not contain the target attribute, then composed values are 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 are 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 are 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 does 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 is a single-valued property with a default of `preserve-all-values`. The available options include:

`preserve-all-values`

Indicates that the server preserves 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 attempts to remove any values for the target attribute that would have been generated for that entry by the associated value patterns but preserves any other values that might 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 are preserved to avoid violating the schema.

`remove-all-values-but-preserve-all-if-required`

Indicates that the server attempts to remove all values for the target attribute from the entry, regardless of whether they would have been composed but falls back to a behavior of `preserve-all-value`s 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 attempts to remove all values for the target attribute from the entry, regardless of whether they would have been composed but falls back to a behavior of `preserve-non-composed-values` if the target attribute type is required by any of the entry's object classes.

Include and exclude criteria configuration

Attribute values are composed by default for every entry in any public backend that contains all of the source attributes used in the appropriate value patterns. To restrict composed values to a specific set of entries, use the `include-base-dn`, `exclude-base-dn, include-filter`, and `exclude-filter` properties.

If the `include-base-dn` property is provided, then values are composed for entries at or below one of those base DNs. If the `exclude-base-dn` property is provided, then values are not composed for any entries at or below one of those base DNs. If both properties are provided and a given entry follows both an include and an exclude base DN, then the exclude base DN takes precedence and values are not generated for that entry. If neither property is specified, then the naming contexts of all public backends are used as the default set of include base DNs.

If the `include-filter` property is provided, then values are composed for entries that match at least one of those filters. If the `exclude-filter` property is provided, then values are not 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 takes precedence and values are not generated for that entry. If neither property is specified, then the server behaves as if an include filter of "&" had been specified.

> **Note**
>
> No virtual attributes are considered when evaluating this filter although composed values might be taken into consideration. For more information about the order in which composed attribute definitions are evaluated, see Composed attribute dependency considerations.

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 `modifyDN` operations.

Existing entries are not automatically populated with generated values. To populate existing entries with generated values, export the data to LDIF and then re-import it. This requires temporarily taking each instance out of service, since importing into a single instance has no effect on the data in other instances. The LDIF import option is 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. The populate composed attribute values 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 distinguished names (DNs) of the composed attribute plugin configuration entries for which values should be generated.

> **Note**
>
> If this attribute is omitted, then values are generated for all defined and enabled composed attribute plugin instances.

`ds-task-populate-composed-attribute-backend-id`

An optional, multi-valued attribute that can be used to specify the backend IDs of the backends in which values should be generated.

> **Note**
>
> If this attribute is omitted, then an appropriate set of backends is 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.

> **Note**
>
> If this attribute is omitted, then no rate limit is 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 uses internal modify operations to make any updates to the entry. These changes are replicated to other servers in the topology, so the task only needs to be invoked on a single instance. However, you should only run it after the composed attribute plugins have been configured on all instances in the topology because this ensures that no entries, including those added or updated while the task is running, are overlooked.

To invoke the task and monitor its progress, use the `populate-composed-attribute-values` command-line tool. This processing is not expected to be needed on a regular basis, so it is not offered as a recurring task.

Composed attribute dependency considerations

Composed attributes must not have dependencies on virtual attributes.

The server does not use virtual attribute values when generating values from a value pattern, nor does it use virtual attribute values when determining if an entry matches an include or exclude filter.

In general, there is no need to have one composed attribute depend on another. If a composed attribute's value does depend on another composed attribute in the entry, the dependent composed attribute can include the value pattern from the source composed attribute as part of its own value pattern. However, this might not be sufficient if clients can override the composed values.

If there is a reason that one composed attribute needs to depend on another, then ensure that the following requirements are satisfied:

- Any composed attribute plugins that have dependencies should be configured so that they are invoked in the appropriate order.

  > **Tip**
  >
  > To configure the order in which composed attribute plugins are invoked, use `plugin-order-ldif-import`.

- Run the populate composed attribute values task 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, you must perform additional runs of the task.
- Enable the `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, 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 fails with an `objectClassViolation (65)` result. This restriction does not exist for operational attributes.

The server allows composed attributes to satisfy mandatory attribute requirements. If the target attribute is declared as a `MUST` type for any of the entry's object classes, then a client should be able to add an entry that does not include values for that attribute type as long as the server composes a value for that attribute.

The server enforces attribute syntax restrictions for composed attributes. If a composed attribute violates the syntax for the associated attribute type, then the operation resulting in that composed attribute value is rejected with an `INVALID_ATTRIBUTE_SYNTAX` result. This can 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 might 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, you cannot configure the composed attribute plugin to generate multiple values for that attribute, and any operation that results in multiple values for the target attribute is rejected.

Composed attributes cannot 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. When 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 restricted in a way that prevents 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 `modifyDN` operation that alters source attributes in a way that results in composed attribute changes.

Synchronization server considerations

Changes involving composed attributes should be synchronized from a PingDirectory server instance to another endpoint type without issues. Changes that create or update composed attributes should be written to the LDAP changelog, just like changes to regular attributes.

Synchronization to PingDirectory server configured with one or more composed attribute plugin instances might be more problematic. It might not be possible to synchronize 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 PingDirectory server is used as a sync destination in an environment in which some attribute values might need to be composed, then you should compose those values in the Synchronization Server using a constructed attribute mapping rather than attempt to generate them in the PingDirectory server with the composed attribute plugin.

PingDirectoryProxy server considerations

Using the PingDirectoryProxy server should not 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, the diagnostic message is returned to the client and included in the access log.

Examples of rejected operations that result in a diagnostic message include:

- If an `add`, `modify`, or `modifyDN` operation is rejected because the composed value violated the syntax for the target attribute type.
- If an attempt was made to alter a composed attribute value in a way that is not permitted by the plugin.

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 can 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.

This is not typically a significant concern, and you can address it 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 that might also be addressed by virtual attribute support. The composed attribute solution might improve performance, searchability, and write behavior, but it also has limitations relative to the virtual attribute subsystem.

Limitations of composed attributes relative to virtual attributes include:

- When creating a composed attribute plugin in the server, you must use manual action 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 are no longer present. When removing or disabling a composed attribute plugin, previously generated composed values remain.
- Virtual attributes can be configured so that the values are only generated under certain conditions, such as only for certain clients, and to enable different values being generated for different clients. Composed attributes are 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 exist in the database, so they 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

This section covers how to configure the PingDirectory server to protect sensitive information in the server.

Some of the ways to protect sensitive data covered in this section are enabling on-disk encryption for data in backends, the changelog, and the replication databases. You can protect sensitive attributes by limiting the ways that clients can interact with them.

For more information, see the following topics:

- About encrypting and protecting sensitive data
- About backing up and restoring the encryption settings definitions
- Configuring sensitive attributes
- Configuring global sensitive attributes
- Excluding a global sensitive attribute on a client connection policy

About encrypting and protecting sensitive data

The PingDirectory 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
- in the changelog
- in the replication databases

You can protect sensitive attributes by limiting the ways that clients can interact with them.

About the encryption settings database

The encryption settings database is a repository that holds information for encrypting and decrypting data.

The database contains encryption settings definitions that specify information about the cipher transformation and encapsulate the key used for encryption and decryption. Before enabling data encryption, you must create an encryption settings definition. An encryption settings definition specifies the cipher transformation to use to encrypt the data and encapsulates the encryption key.

You can use the `encryption-settings` tool to manage the encryption settings database, including:

- Creating, deleting, exporting, and importing encryption settings definitions
- Listing the available definitions
- Indicating which definition to use for subsequent encryption operations
- Managing data encryption restrictions to impose on the server
- Freezing and unfreezing the encryption settings database
- Supplying the passphrase needed for the Wait for Passphrase cipher stream provider

For more about the `encryption-settings` tool, see Using the encryption-settings tool.

Implementing encryption settings definitions

Although the encryption settings database can have multiple encryption settings definitions, you must designate only one of them as the preferred definition. The preferred encryption settings definition is used for all subsequent encryption operations. Any existing data that has not yet been encrypted remains unencrypted until it is rewritten, such as a result of a `modify` or `modifyDN` operation, or if the data is exported to LDIF and re-imported.

If you introduce a new preferred encryption settings definition, then any existing encrypted data continues to use the previous definition until it is rewritten. If you do change the preferred encryption settings definition for the server, keep the previous definitions in the database until you have verified that no remaining data uses the older keys.

Supported encryption ciphers and transformations

Learn how to implement encryption ciphers, cipher algorithms, and cipher stream providers in the PingDirectory server.

Cipher algorithms

The PingDirectory server supports encryption cipher suites that are compliant with the Java Virtual Machine (JVM) on which the server is running. When configuring encryption, you must specify the cipher using a key length in bits and either a cipher algorithm name, such as AES, or a full cipher transformation that explicitly specifies the mode and padding to use for the encryption, such as AES/CBC/ PKCS5Padding. If only a cipher algorithm is given, then the default mode and padding for that algorithm are automatically selected.

The following cipher algorithms and key lengths have been tested using the Oracle JVM.

Cipher algorithms and key length

Cipher Algorithm	Key Length (bits)
AES	128
Blowfish	128
DES	64
DESede	192
RC4	128

Tip

Although the server supports all of the cipher algorithms listed in the table, using any algorithm other than AES is discouraged. The DES and RC4 algorithms have known vulnerabilities, and there are no meaningful advantages in using Blowfish or DESede instead of AES.

By default, some JVM implementations come with limited encryption strength, which might restrict the usable key lengths. For example, the Oracle JVM does not allow AES with 192-bit or 256-bit keys unless you download and install the unlimited encryption strength policy files.

Cipher modes indicate how the cipher algorithm should be recurrently applied to portions of the data to produce the final result. Supported cipher transformation modes include:

CBC

The Cipher Block Chaining mode uses an initialization vector to introduce randomness into the encryption process. This ensures that encrypted outputs are always unique even when the input data is identical.

GCM

The Galois/Counter mode provides better data integrity protection to ensure that ciphertexts are not altered. Like the CBC mode, GCM uses an initialization vector to introduce randomness into the output.

ECB

The Electronic Codebook mode encrypts each block of data separately without using any randomness. Because of the lack of randomness, this mode is susceptible to plain-text attacks.

Padding is important when using a block cipher because these ciphers can only encrypt a fixed amount of data at a given time. If the amount of data to be encrypted is not a multiple of the block size, then extra bytes must be added to the plain text before encrypting it. It must also be possible to remove these extra bytes when decrypting. Supported padding algorithms in a cipher transformation include:

PKCS5Padding

This is a standard padding algorithm described in the PKCS #7 specification in RFC 5652. You should use this padding algorithm when using a block cipher like AES with the CBC or ECB modes. Optionally, you can also use this algorithm when using the AES block cipher with the GCM mode.

NoPadding

This algorithm indicates that no padding is needed and should be used with stream ciphers rather than block ciphers.

> **Note**
>
> AES can act as a stream cipher when using the GCM mode.

For specific reference information about the algorithms and transformations available in all compliant JVM implementations, see the Java Cryptography Architecture Reference Guide and Java Cryptography Architecture Standard Algorithm Name Documentation documentation.

Cipher stream providers

By default, setup generates a strong, random passphrase and writes it to a file. The server then uses a file-based cipher stream provider to read the passphrase and generate a key for encrypting the contents of the encryption settings database. However, the server supports additional cipher stream providers that 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.
- Use a key generated from a passphrase stored in the Amazon Secrets Manager service.
- Use a key generated from a passphrase stored in the Azure Key Vault service.
- Use a key generated from a passphrase stored in a CyberArk Conjur instance.
- Use a key generated from a certificate stored in a PKCS #11 token.

For more information, see Configuring cipher stream providers.

Using the encryption-settings tool

The `encryption-settings` tool provides a mechanism for interacting with the server's encryption settings database.

About this task

Use the `encryption-settings` tool to:

- List the available encryption settings definitions.
- Create new encryption settings definitions.
- Delete existing encryption settings definitions.
- Indicate which encryption settings definition is the preferred definition.
- Export encryption settings definitions to a file for backup purposes and to allow them to be imported for use in other PingDirectory server instances.

- Enable and disable data encryption restrictions for the server and list active restrictions.
- Freeze or unfreeze the encryption settings database.
- Supply the passphrase for the Wait for Passphrase cipher stream provider to unlock the encryption settings database.

Steps

- To display the set of available encryption settings definitions, use the `encryption-settings` tool with the `list` subcommand.

  This subcommand does not take any arguments.

  Example:

  ```
  $ bin/encryption-settings list
  ```

  Result:

  For each definition, the result includes:

  - The unique identifier for the definition
  - Whether the definition is the preferred definition
  - The cipher transformation and key length that are used for encryption

  ```
  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

The `encryption-settings` tool with the `create` subcommand provides a mechanism for creating a new encryption settings definition.

Steps

- To create an encryption settings definition, use the `encryption-settings` tool with the `create` subcommand.

  This subcommand takes the following arguments.

Argument	Description
`--cipher-algorithm <algorithm>` (required)	Specifies the base cipher algorithm to use. Make sure the *<algorithm>* input is the name of the algorithm, such as AES.
`--cipher-transformation <transformation>` (optional)	Specifies the full cipher transformation to use including the cipher mode and padding algorithms, such as AES, CBC, and PKCS5Padding. If you do not provide this argument, a default transformation is used for the specified cipher algorithm.
`--key-length-bits <length>` (required)	Specifies the length of the encryption key in bits, such as 128.
`--set-preferred`	Indicates that the new encryption-settings definition is made the preferred definition and used for subsequent encryption operations in the server. By default, the first definition you create in the encryption settings database is the preferred definition.
`--key-factory-iteration-count <count>`	Specifies the PBKDF2 iteration count to be used when deriving the encryption key for the resulting definition.
`--prompt-for-passphrase`	Indicates that the tool should prompt the user for a passphrase to use in generating the encryption settings definition.
`--passphrase-file <path>`	Specifies the file path containing the passphrase to be used in generating the encryption settings definition.
`--description <description>`	Provides a readable string describing the purpose of the encryption settings definition.

Each encryption settings definition is backed by a passphrase that generates an associated encryption key. You can specify the passphrase when creating the definition by typing it into an interactive prompt or providing it in a file. If you do not specify a passphrase, then the `encryption-settings` tool will generate a strong random passphrase.

The created encryption settings definition has an identifier that the server uses to indicate the definition used in encrypting a given block of data. That way, the correct definition can be used to decrypt that data. This identifier is generated from several elements, including:

- The passphrase backing the definition
- The iteration count used to derive the encryption key from the passphrase
- The derived key length
- The cipher algorithm and cipher transformation used to perform the encryption

If you create an encryption settings definition with the same passphrase, iteration count, key length, cipher algorithm, and cipher transformation across multiple servers, then each of the servers will have an identical encryption settings definition. To create a definition without specifying a passphrase, use the `encryption-settings export` command to export that definition from its native server instance. Then, use the `encryption-settings import` command to import the definition into other instances of the topology.

> **Note**
>
> Although the `--key-factory-iteration-count` argument is optional, you should provide this argument whenever generating a new encryption settings definition because the default iteration count is lower than the minimum count of 600,000 recommended by OWASP.

Example:

```
$ bin/encryption-settings create --cipher-algorithm AES \
--cipher-transformation AES/GCM/NoPadding \
--key-factory-iteration-count 600000 \
--key-length-bits 256 \
--set-preferred \
--description "Example encryption settings definition with a randomly generated key"
```

Result:

```
Successfully created a new encryption settings definition with ID
F635E109A8549651025D01D9A6A90F7C9017C66D
```

Changing the preferred encryption settings definition

The `encryption-settings` tool with the `set-preferred` subcommand provides a mechanism for changing the preferred encryption settings definition.

About this task

To change a definition to a preferred definition:

Steps

- To change a definition, use the `encryption-settings` tool with the `set-preferred` subcommand to change a definition to a preferred definition.

  This subcommand takes the following argument.

The `set-preferred` subcommand accepted argument

Argument	Description
`--id <id>` (required)	Specifies the ID to export for the encryption settings definition.

Example:

```
$ bin/encryption-settings set-preferred --id 4D86C7922F71BB57B8B5695D2993059A26B8FC01
```

Result:

```
Encryption settings definition 4D86C7922F71BB57B8B5695D2993059A26B8FC01 was
successfully set as the preferred definition for subsequent encryption operations
```

Deleting an encryption settings definition

To free space in the encryption settings database, you can use the `encryption-settings` tool to delete encryption settings definitions.

About this task

You should not remove an encryption settings definition that the server is currently using because it will no longer be possible to access any data encrypted by the removed definition. In some cases, removing a definition used to encrypt live data in the database (which can include local DB backends, the replication database, or the LDAP changelog) prevents the server from starting or accessing content in the backend.

> **Note**
>
> Do not remove encryption settings definitions unless there is reason to believe they are compromised. If you believe a key has been compromised, see Handling compromised encryption settings definitions for details on safely removing that key.

To delete an encryption settings definition:

Steps

- Use the `encryption-settings` command with the `delete` subcommand.

> **Important**
>
> Make sure to include the `--id` argument to specify the definition.

Argument	Description
`--id <id>` (required)	Specifies the ID of the encryption settings definition to delete.

Example:

```
$ bin/encryption-settings delete --id F635E109A8549651025D01D9A6A90F7C9017C66D
```

Result:

```
Successfully deleted encryption settings definition
F635E109A8549651025D01D9A6A90F7C9017C66D
```

Configuring data encryption restrictions

The PingDirectory server supports several data encryption restrictions that make it harder for unauthorized individuals to access data in an unencrypted form.

About this task

> **Note**
>
> By default, none of the available data encryption restrictions are active in the server.

Steps

- To configure data encryption restrictions, use the `encryption-settings set-data-encryption-restrictions` command with one of the following arguments.

Arguments	Description
`--add-restriction <restriction-name>`	Activates the specified encryption restriction in the server. You can provide this argument multiple times with a single command to add multiple restrictions.
`--remove-restriction <restriction-name>`	Removes the specified encryption restriction from the server. You can provide this argument multiple times with a single command to remove multiple restrictions.
`--remove-all-restrictions`	Removes any data encryption restrictions that are currently in place.
`--add-all-restrictions`	Activates all supported data encryption restrictions that are not already active.

Example:

```
$ bin/encryption-settings set-data-encryption-restrictions \
    --add-all-restrictions
```

After the successful completion of the previous command, you receive a message like the following:

```
Successfully updated the set of active data encryption restrictions.

The updated set of active data encryption restrictions is:
* prevent-disabling-data-encryption.
* prevent-changing-cipher-stream-provider.
* prevent-encryption-settings-export.
* prevent-unencrypted-ldif-export.
* prevent-passphrase-encrypted-ldif-export.
* prevent-unencrypted-backup.
* prevent-passphrase-encrypted-backup.
* prevent-decrypt-file.
```

- To determine which data encryption restrictions are active in the server, use the `encryption-settings get-data-encryption-restrictions` command.

> **Note**
>
> If you are defining data encryption restrictions in the server, freeze the encryption settings database so that these restrictions cannot be modified by anyone without the appropriate passphrase. For more information, see Freezing the encryption settings database.

Freezing the encryption settings database

You can freeze the encryption settings database with a specified passphrase. While it is frozen, the database operates in read-only mode.

About this task

If the encryption settings database is frozen, the server can use the database for data encryption processing but will not allow any of the following:

- Creating new encryption settings definitions
- Importing encryption settings definitions from an exported set
- Removing encryption settings definitions
- Specifying which definition is preferred for new encryption operations
- Adding or removing data encryption restrictions

To make changes to a frozen database, you must unfreeze it by providing the passphrase originally used to freeze it.

Steps

- To freeze the encryption settings database, use the `encryption-settings freeze` command.

  This command supports the `--passphrase-file <path>` argument, which specifies the path to a file containing the passphrase to use for freezing the encryption settings database. If the argument is not provided, the `encryption-settings freeze` command prompts the user for the passphrase.

  Example:

```
$ bin/encryption-settings freeze
Enter the passphrase to use to freeze the encryption settings database:
Confirm the freeze passphrase:
Successfully froze the encryption settings database.
```

- To unfreeze the encryption settings database, use the `encryption-settings unfreeze` command and provide the passphrase originally used to freeze the database.

  This command supports the `--passphrase-file <path>` argument, which specifies the path to a file containing the passphrase to use for unfreezing the encryption settings database. If this argument is not provided, the `encryption-settings unfreeze` command prompts the user for the passphrase.

  Example:

  ```
  $ bin/encryption-settings unfreeze
  Enter the passphrase used to freeze the encryption settings database:
  Successfully unfroze the encryption settings database.
  ```

- To determine whether the encryption settings database is currently frozen, use the `encryption-settings is-frozen` command.

  This command does not require any arguments.

  Example:

  ```
  $ bin/encryption-settings is-frozen
  The encryption settings database was frozen at Mon Mar 06 22:42:10 UTC 2023.
  ```

Encrypting and decrypting files

You can use the `encrypt-file` tool to encrypt and decrypt files with an encryption settings definition or with a supplied passphrase.

About this task

When a file is encrypted with an encryption settings definition, the server can automatically determine that the file is encrypted, retrieve the associated definition from the encryption settings database, and use it to access the file's contents.

Encrypting a file with an encryption settings definition is useful for files containing sensitive content needed for processing. Examples include:

- PIN files for certificate keys and trust stores
- The `tools.properties` file that contains default arguments for command-line tools
- Bind password files for command-line tools
- Files used for file-based passphrase providers

> **Note**
>
> The server does not support encrypting the configuration or schema files. It also does not support encrypting files needed by the configured cipher stream provider to access the encryption settings database.

To encrypt a file with the server's preferred encryption settings definition:

Steps

- **Use the `encrypt-file` tool.**

  Example:

  ```
  $ bin/encrypt-file --input-file password.txt \
  --output-file password.txt.encrypted
  ```

  The `encrypt-file` tool can also decrypt the results of encrypted output files generated by the server, including encrypted backups, LDIF exports, and log files. However, this decryption cannot be performed if the `prevent-decrypt-file` data encryption restriction is active.

  Useful arguments to use with the `encrypt-file` tool include.

Arguments	Description
`--input-file <path>`	Specifies the path to the file containing the plain-text data to be encrypted. If you do not provide this argument, then the data will be read from standard input.
`--output-file <path>`	Specifies the path to the file to which the encrypted data should be written.
`--decrypt`	Indicates that the data in the input file should be decrypted rather than encrypted. Use of this argument is not allowed if the `prevent-decrypt-file` data encryption restriction is enabled.
`--encryption-settings-id <id>`	Specifies the ID associated with the encryption settings definition to be used in encrypting the input file. By default, the server uses the preferred encryption settings definition.
`--prompt-for-passphrase`	Indicates that the tool should prompt the user for a passphrase to use to encrypt the file, rather than encrypting the file with an encryption settings definition. The server cannot automatically decrypt passphrase-encrypted files.
`--passphrase-file <path>`	Specifies the path to the file containing the passphrase to use to encrypt the file.

Arguments	Description
`--compress-output`	Indicates that the server should gzip-compress the output. When encrypting data, the output is compressed before it is encrypted. When decrypting data, the data is compressed after it is decrypted.
`--decompress-input`	Indicates that the input file is gzip-compressed. When decrypting data, the data is decompressed after it is decrypted.

Use the `encrypt-file --help` command to see a complete set of arguments supported by the `encrypt-file` tool.

About backing up and restoring the encryption settings definitions

The PingDirectory server provides two different mechanisms for backing up and restoring encryption settings definitions.

To back up and restore encryption settings definitions, you can either:

- Export one or more encryption settings definitions using the `encryption-settings export` command. This command also generates a passphrase-protected file containing the encryption settings definitions in a portable format. This is the recommended approach.
- Back up and restore the entire encryption settings database using the `backup` and `restore` tools.

The `encryption-settings export` command is recommended for the following reasons:

- With the `backup` command, the resulting backup only contains the `encryption-settings-db` file. The backup does not automatically contain any metadata files needed by the configured cipher stream provider to access the encryption settings database. The output generated by the `backup` command indicates which additional files are needed to enable that access.

  > **Note**
  >
  > These metadata files must be present before restoring the encryption settings database.

- With the `backup` command, the resulting backup depends on the cipher stream provider enabled when the encryption settings database was last written. This means that the cipher stream provider must already be configured and active in the server configuration before restoring the encryption settings database.
- Because the backup generated by the `backup` command depends on the existing configuration of the cipher stream provider, that cipher stream provider might not be useable if the system configuration changes. For example, if the Amazon Key Management Service cipher stream provider was used at the time the backup was generated with an encryption key that is no longer available, then it's not possible to restore the backup.

> **Important**
>
> The encryption settings definitions must be exported or backed up regularly, especially after creating a new definition, importing one or more definitions, or changing the preferred encryption settings definition.
> If an encryption settings definition is lost, then any data encrypted with that definition becomes inaccessible. In some cases, a lost definition renders the server inoperable.

Exporting encryption settings definitions

Use the `encryption-settings` tool with the `export` subcommand to export encryption settings definitions.

About this task

The `encryption-settings export` command creates a portable, passphrase-protected export of one or more encryption settings definitions. You can use encryption settings exports in the following ways:

- As the preferred method for backing up encryption settings definitions. The export format is portable, does not depend on the cipher stream provider configuration, and can be used across server versions.
- As a way to transfer encryption settings definitions between servers.
- As a way to set up new server instances with an appropriate set of definitions. When executing `setup`, you can use the `--encryptDataWithSettingsImportedFromFile` and `--encryptionSettingsExportPassphraseFile` options to enable encryption with definitions from an export file.

Steps

- To export the encryption settings definitions to a file, use the `encryption-settings` tool with the `export` subcommand.

  The subcommand can take the following arguments.

Arguments	Description
`--id <id>`	Specifies the ID to export for the encryption settings definition. You can specify this argument multiple times. If it's omitted, all definitions are exported.
`--output-file <path>` (required)	Specifies the path to the output file to write the encryption settings definition to.
`--pin-file <path>`	Specifies the path to a passphrase file containing the password for encrypting the contents of the exported definition. If this argument isn't provided, then the PIN is interactively requested.

  Example:

  The following example shows the specific path to an output file for the exported encryption settings definition:

```
$ bin/encryption-settings export --output-file /tmp/exported-key
Enter the PIN to use to encrypt the definition:
Re-enter the encryption PIN:
Successfully exported encrpytion settings data to file /tmp/exported-key
```

  The successful export returns the following:

```
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 isn't provided, then the PIN is 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.

  Example:

  ```
  $ bin/encryption-settings import --input-file /tmp/exported-key --set-preferred
  Enter the PIN used to encrypt the definition:
  ```

  Result:

  ```
  Successfully imported encryption settings definition
  F635E109A8549651025D01D9A6A90F7C9017C66D from file /tmp/exported-key
  ```

Enabling data encryption in the server

Using data encryption ensures that all future operations written to the local backends, LDAP changelog, and replication database are protected from unauthenticated sources.

About this task

Enabling data encryption after setting up the server can result in unencrypted data being stored in local DB backends, the LDAP changelog, and the replication database. You should export all local DB backends to LDIF and re-import the data after enabling encryption to ensure that all data is properly encrypted. This will also ensure that all indexes are encrypted.

To enable data encryption:

Steps

- Use the `dsconfig` tool with the `set-global-configuration-prop` subcommand and set one of the following properties.

Global property	Configuration description
`encrypt-data`	Indicates whether data encryption is enabled in the server: ◦ If set to `true`, then subsequent writes to the local DB backends, the replication database or the LDAP changelog will be encrypted. ◦ If set to `false`, then subsequent writes will be stored in unencrypted form.
`encryption-settings-cipher-stream-provider`	Indicates which cipher stream provider should be used to protect the contents of the encryption settings database. By default, an instance of the file-based cipher stream provider is used to encrypt the database with a key generated by a passphrase read from a specified file.
`encrypt-backups-by-default`	Indicates whether the server should encrypt backups by default: ◦ If set to `true`, a specified `backup-encryption-settings-definition-id` value is used to generate the encryption key for the backup. ◦ If a value is not specified, the server attempts to use the preferred encryption settings definition to generate the encryption key. ◦ If the server is not configured with any encryption settings definitions, it uses an internal key shared across instances in the topology. ◦ You can override the property's value with the `backup` tool and either the `--encrypt` or `--doNotEncrypt` option.

Global property	Configuration description
`backup-encryption-settings-definition-id`	Specifies the unique identifier of the encryption settings definition to use in generating the encryption key: ◦ 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. ◦ You can override the property's value with the **`backup`** tool. Using the `--promptForEncryptionPassphrase` or `--encryptionPassphraseFile` option will generate the encryption key from the provided passphrase. The `--encryptionSettingsDefinitionID` option will generate the key from a specified encryption settings definition.
`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 a value is not specified, the server first tries to use the preferred encryption settings definition to generate the encryption key. ◦ If the server is not configured with any encryption settings definitions, it uses an internal key shared among instances in the topology. ◦ You can override the property's value with the **`export-ldif`** tool and either the `--encryptLDIF` or `--doNotEcryptLDIF` option.
`automatically-compress-encrypted-ldif-exports`	Indicates whether to automatically compress LDIF exports that are encrypted. If set to `true`, any LDIF export that is encrypted, either with the `--encryptLDIF` option or with the **`encrypt-ldif-exports-by-default`** configuration property, is gzip-compressed.

Example:

```
$ bin/dsconfig set-global-configuration-prop --set encrypt-data:true
```

Using data encryption in a replicated environment

Use data encryption for on-disk storage for data within the server.

Whenever clients access stored data in the server, it's presented in unencrypted form although the communication with those clients can itself be encrypted using SSL or StartTLS. Replication, the communication of updates between replication servers, is always encrypted using SSL. Each server can apply data encryption in a independent manner and have different sets of encryption settings definitions. It's possible to have a replication topology containing some servers that have data encryption enabled and others with it disabled.

When initializing the backend of one server from another server with data encryption enabled, the server being initialized must have access to all encryption settings definitions that might 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`.

Handling compromised encryption settings definitions

When an encryption settings definition is compromised, all data encrypted with that definition is vulnerable and you must stop using the definition immediately.

About this task

If an encryption settings definition is compromised, stop using that definition immediately. You must re-encrypt any data encrypted with the compromised definition using a new definition or purge that data from the server. To minimize the risk of data exposure, act quickly on all servers using this definition and act on one server at a time to avoid environment-wide downtime.

> **Important**
>
> Before removing the compromised encryption settings definition, you should run the **`encrypt-file --find-encrypted-files`** command to search for encrypted files on the server. If any files are encrypted with a key tied to the compromised encryption settings definition, those files will no longer be accessible after you remove the definition, potentially preventing the server from starting or from functioning properly. If any encrypted files are found, run **`encrypt-file --re-encrypt`** to re-encrypt the files with a different definition before removing the compromised definition.

If you have a compromised encryption settings definition:

Steps

1. Create a new encryption settings definition and make it the preferred definition for new write operations.
2. Ensure that client traffic is routed away from the compromised server instance.

   If the PingDirectory server is accessed through a PingDirectoryProxy server, then you can set the `health-check-state` configuration property for any LDAP external server definitions that reference that server to `unavailable`.

3. To ensure that external clients are not allowed to perform writes in the PingDirectory server, set the `writability-mode` global configuration property to `internal-only`.
4. Look 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.

   This signals that all outstanding local changes are replicated to other servers.
5. Stop the PingDirectory server instance.
6. Delete the replication server database by removing all files in the `<server-root>/changeLogDb` directory.

   If all local changes have been replicated to other servers, this deletion does not result in any data loss in the replication environment.
7. Export the contents of all local DB and changelog backends to LDIF.

   > **Note**
   >
   > If you are using the AES256 scheme to store passwords in a reversibly encrypted form, and if any of those passwords are encrypted with the compromised encryption settings definition, you should use the **`export-reversible-passwords`** tool rather than the **`export-ldif`** tool to perform the LDIF export of the local DB backends. You can use the **`export-reversible-passwords`** tool to generate an LDIF file in which all reversibly encrypted passwords are exported in a decrypted form so that when they are re-imported, they are re-encrypted.
8. Re-import the data from LDIF.

   The data is now encrypted using the new preferred encryption settings definition.
9. Use the **`encrypt-file --find-encrypted-files`** command to find any files that are encrypted with the compromised definition, and use **`encrypt-file --re-encrypt`** to re-encrypt them with the new definition.
10. Export the compromised encryption settings definition from the encryption settings database.

    As a precaution, this backs up the definition in case some remaining data was encrypted with that definition's key.
11. To prevent the PingDirectory server from using the compromised definition, delete the definition from the encryption settings database.
12. Start the PingDirectory server instance.
13. Allow replication to update the server with any changes processed offline.
14. To re-allow externally-performed write operations, change the value of the global `writability-mode` configuration property to `enable`.
15. To allow client traffic to re-route to that server instance, reconfigure the environment.

    For example, if you changed the value of the `health-check-state` property in step 2, change it back to `dynamically-determined`.

Configuring sensitive attributes

Use and configure sensitive attribute definitions to customize the level of client data access and encryption protection needed.

For a PingDirectory server instance, data encryption only applies to the on-disk storage. Although it doesn't automatically protect information accessed or replicated between servers, the server offers mechanisms to provide that protection, such as SSL, StartTLS, and SASL.

All client communication using either SSL or StartTLS encryption and all replication traffic using SSL encryption helps ensure that the data is protected from unauthorized individuals who might eavesdrop on network communication. You can enable this communication security independently of data encryption.

> **Note**
>
> If data encryption is enabled, you should use secure communication to protect network access to that data.

Protecting client data access isn't as simple as enabling secure communication. In some cases, you might want to allow insecure access to some data. In other cases, you might need to have additional levels of protection in place to ensure that some attributes are more carefully protected. To achieve varying levels of protection, use sensitive attribute definitions.

The following table explains the sensitive attribute definitions and their configuration properties.

Sensitive attribute	Configuration details
`attribute-type`	Specifies the set of attribute types whose values might be considered sensitive. You must provide at least one attribute type and define all specified attribute types in the server schema.
`include-default-sensitive-operational-attributes`	Indicates whether the set of sensitive attributes that might contain sensitive information should automatically be updated to include any operational attributes maintained by the PingDirectory server itself. This includes the `ds-sync-hist` operation attribute, which is used for data required for replication conflict resolution and can contain values from other attributes in the entry.

Sensitive attribute	Configuration details
`allow-in-filter`	Indicates whether sensitive attributes can be used in filters. This applies not only to the filter used in search requests, but also to filters that can be used in other places, such as the assertion and join request controls. The value of this property must be one of the following: • `Allow` : allows sensitive attributes to be used in filters over both secure and insecure connections. • `Reject` : rejects any request that includes a filter targeting one or more sensitive attributes over both secure and insecure connections. • `Secure-only` : allows sensitive attributesto be used in filters over secure connections but rejects any such requests over insecure connections.
`allow-in-add`	Indicates whether sensitive attributes can be included in entries created by `ldapadd` operations. The value of this property must be one of the following: • `Allow` : allows sensitive attributes included in add requests over both secure and insecure connections. • `Reject` : rejects any add request containing sensitive attributes over both secure and insecure connections. • `Secure-only` : allows sensitive attributes included in add requests received over secure connections but rejects any such requests over insecure connections.
`allow-in-compare`	Indicates whether sensitive attributes can be targeted by the assertion used in a compare operation. The value of this property must be one of the following: • `Allow` : allows sensitive attributes to be targeted by requests over both secure and insecure connections. • `Reject` : rejects compare requests targeting a sensitive attribute over both secure and insecure connections. • `Secure-only` : allows compare requests targeting sensitive attributes over secure connections but rejects compare requests over insecure connections.

Sensitive attribute	Configuration details
`allow-in-modify`	Indicates whether sensitive attributes can be updated using modify operations. The value of this property must be one of following: • `Allow` : allows sensitive attributeto be modified by requests over both secure and insecure connections. • `Reject` : rejects any modify request updating a sensitive attribute over both secure and insecure connections. • `Secure-only` : allows only modify requests updating sensitive attributes over secure connections but rejects modify requests over insecure connections.

Note

By default, `allow-in-returned-entries` , `allow-in-filter` , `allow-in-add` , `allow-in-compare` , and `allow-in-modify` properties have values of `secure-only` only. This prevents the possibility of exposing sensitive data in the clear to anyone able to observe network communication.

Sensitive attributes and client connection policies

If a client connection policy references a sensitive attribute definition, any restrictions imposed by that definition are enforced for clients associated with this client connection policy. If multiple sensitive attribute definitions are associated with a client connection policy, the server uses the most restrictive combination of all of those sets.

Sensitive attributes and other security mechanisms

The sensitive attribute definitions work in conjunction with other security mechanisms defined in the server and can only be used to enforce additional restrictions on clients.

Never use sensitive attribute definitions to grant a client additional access to information that it didn't have already through other means. For example, if the `employeeSSN` attribute is declared a sensitive attribute and the `allow-in-returned-entries` property has a value of `Secure-only` , then the `employeeSSN` attribute is only returned to those clients that have both permissions granted by the access control rules defined in the server and are communicating with the server over a secure connection.

The `employeeSSN` attribute is stripped out of entries returned to clients normally authorized to see it if they are using insecure connections. It is also stripped out of entries for clients normally not authorized to see it even if they have established secure connections.

Creating a sensitive attribute

Creating sensitive attributes involves creating and associating definitions to set configuration properties and policies according to your business needs.

About this task

To create a sensitive attribute:

Steps

1. Create one or more sensitive attribute definitions using `dsconfig create-sensitive-attribute`.

   Example:

   For example, to create a sensitive attribute definition that only allows access to the `employeeSSN` attribute by clients using secure connections, make the following configuration changes.

   ```
   $ 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 the sensitive attribute definitions with the client connection policies that you want to enforce using `dsconfig set-client-connection-policy-prop`.

   Example:

   ```
   $ bin/dsconfig set-client-connection-policy-prop --policy-name default \
     --set "sensitive-attribute:Employee Social Security Numbers"
   ```

Configuring global sensitive attributes

The PingDirectory server supports the ability to define sensitive attributes as a global configuration option so that they're automatically used across all client connection policies.

About this task

Administrators can assign one or more sensitive attribute definitions to a client connection policy.

> **Note**
>
> When working in an environment with multiple client connection policies, it can be easy to add a sensitive attribute definition to one policy but overlook it in another.

Steps

- To add a global sensitive attribute across all client connection policies, run the `dsconfig` tool.

  Example:

  The following command adds the `employeeSSN` as a global sensitive attribute, which is applied across all client connection policies.

```
$ bin/dsconfig set-global-configuration-prop --add "sensitive-attribute:employeeSSN"
```

Excluding a global sensitive attribute on a client connection policy

Administrators can exclude a global sensitive attribute on a client connection policy when it's not needed for client connection requests.

About this task

Administrators can set a global sensitive attribute across all client connection policies. However, there can be cases when a specific PingDirectory server must exclude the sensitive attribute because it's not needed for client connection requests.

For example, in most environments, it's good to declare the `userPassword` attribute to be a sensitive attribute that prevents external clients reading it. This solution is more secure than protecting the `password` attribute using the server's default global access control instruction (ACI), which only exists for backwards compatibility purposes. If the PingDirectory server is installed, then it does need 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 server. The PingDirectory server provides an `exclude-global-sensitive-attribute` property for this purpose.

Steps

1. To remove the global ACI that limits access to the `userPassword` or `authPassword` attribute, run the **`dsconfig`** tool.

   > **Note**
   >
   > The global ACI is present for backwards compatibility.

   Example:

   ```
   $ 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. To add the `userPassword` attribute as a global sensitive attribute, run the **`dsconfig`** tool and add the built-in `"sensitive-attribute:Sensitive Password Attributes"` definition to the global configuration.

   This applies to all client connection policies.

   Example:

   ```
   $ 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, configure a client connection policy for the Sync User to exclude the global sensitive attribute.

   Example:

The following example shows how to create a new policy if the Data Sync Server binds with the default distinguished name (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 PingDirectory server provides a client-accessible Lightweight Directory Access Protocol (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 PingDirectory 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 used by external software to maintain application compatibility between client services. For example, you can install the PingDirectory server that monitors the LDAP changelog for any updates that occur on a source PingDirectory server and synchronizes these changes to a target directory information tree (DIT) or database server.

The PingDirectory 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 (the external changelog that clients can access) physically resides at `<server-root>/db/changelog`.
> - The Replication changelog backend (the changelog that replication servers use) physically resides at `<server-root>/changelogDB`.

Key changelog features

The PingDirectory server supports changelog backend properties that allow access control filtering and sensitive attribute evaluation for targeted entries.

External client applications can change the contents of attributes seen in the targeted entry based on the access control rules applied to the associated base DN.

Attribute	Description
`apply-access-controls-to-changelog-entry-contents`	Indicates whether the contents of changelog entry attributes, such as `changes`, `deletedEntryAttrs`, `ds-changelog-entry-key-attr-values`, `ds-changelog-before-values`, and `ds-changelog-after-values`, are subject to access control and 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 are 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 are also 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 might have been removed because of 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 operational attributes that have been removed. If any user attribute information was excluded from a changelog entry, the number of the excluded user attributes are 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 are 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 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 are 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 are 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 using `+` to be returned.

Enabling access control filtering in the LDAP changelog

Use the `dsconfig` tool to enable the properties to the changelog backend to set up access control to the LDAP changelog.

About this task

Only admin users with the `bypass-acl` privilege can read the changelog.

Steps

1. To allow LDAP clients to undergo access control filtering using standard LDAP searches of the `cn=changelog` backend, enable the `apply-access-control-to-changelog-entry-contents` property.

   Access control filtering is applied regardless of the value of the `apply-access-controls-to-changelog-entry-contents` setting when the changelog backend is servicing requests from a PingDirectory server that has the `filter-changes-by-user` Sync Pipe property set.

   Example:

   ```
   $ bin/dsconfig set-backend-prop --backend-name "changelog" \
     --set "apply-access-controls-to-changelog-entry-contents:true"
   ```

2. To include a count of users that have been removed through access control filtering, set the `report-excluded-changelog-attributes` property.

   The count appears in the `ds-changelog-num-excluded-user-attributes` attribute for users and in the `ds-changelog-num-excluded-operational-attributes` attribute for operational attributes.

   Example:

   ```
   $ bin/dsconfig set-backend-prop --backend-name "changelog" \
    --set "report-excluded-changelog-attributes:attribute-counts"
   ```

Useful changelog features

The PingDirectory server provides two 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 are 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 and after values are stored in the `ds-changelog-attr-exceeded-max-values-count` attribute on the changelog entry. This is a multi-valued attribute with the following format.:

```
attr=attributeName,beforeCount=100,afterCount=101
```

`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 must reset the `changelog-max-before-after-values` property to a higher value. When this attribute is set, an alert is 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 are not 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 are 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 are recorded in the LDAP changelog.

For example, consider 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
```

Run the `bin/base64 decode -d` command line tool to view the decoded value for the `changes`, `ds-changelog-before-values`, and `ds-changelog-after-values` attributes.

After base64 decoding, the `changes` attribute displays the following.

```
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 displays the following.

```
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 displays the following.

```
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

To view the LDAP changelog properties, run the `dsconfig` `get-backend-prop` command and specify the changelog backend.

Viewing the LDAP changelog properties using `dsconfig` non-interactive mode

Steps

- To view the changelog properties on the PingDirectory server, use `dsconfig`.

> **Tip**
>
> 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.

Example:

```
$ bin/dsconfig get-backend-prop --backend-name changelog
```

Result:

```
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

Enable the LDAP changelog to maintain application compatibility between client services.

The LDAP changelog is disabled by default on the PingDirectory 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**
>
> Enable the feature using the **`dsconfig`** tool only if required because it can affect LDAP update performance.

Enabling the LDAP changelog using `dsconfig` non-interactive mode

Steps

- Use `dsconfig` to enable the changelog property on the PingDirectory server.

  Example:

```
$ 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.
2. Authenticate to the server by entering the host name, LDAP connection, port, bindDN, and bind password.
3. In the `PingDirectory server main menu`, enter `o` to change from the Basic object level to the Standard object level.
4. Enter the option to select the `Standard object` level.
5. In the `PingDirectory server main menu`, enter the number corresponding to `backend`.
6. In the `Backend Management menu`, enter the option to view and edit an existing backend.

   Result:

   The system displays a list of the accessible backends. For example, you see options for the `changelog` and `userRoot` backends.
7. Enter the option to work with the changelog backend.
8. In the `Changelog Backend properties menu`, enter the number corresponding to the Enabled property.
9. In the `Enabled Property menu`, enter the number to change the `Enabled` property to `TRUE`.
10. In the `Backend Properties menu`, enter `f` to apply the change.

    If you set up the server in a server group, enter `g` to update all of the servers in the group. Otherwise, repeat steps 1-10 on the other servers.
11. Verify that changes made to the data are recorded in the changelog.

Changing the LDAP changelog database location

If you have disk space issues, you can change the on-disk location of the LDAP changelog 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 restart the PingDirectory 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`.

   Example:

   The following command sets the LDAP changelog backend to `<server-root>/db2`.

```
$ bin/dsconfig set-backend-prop --backend-name changelog \
  --set "db-directory:db2" --set "enabled:true"
```

> **Note**
>
> Remember to include the LDAP connection parameters, such as host name, port, bindDN, and bindPassword.

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.

   Because the LDAP changelog backend was previous disabled, you do not need to manually relocate any existing database files.

   Example:

```
$ 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, to reset it to its original location, use `dsconfig`.

   Example:

   The following command resets the LDAP changelog backend to `<server-root>/db` location.

```
$ bin/dsconfig set-backend-prop --backend-name changelog \
  --reset "db-directory"
```

> **Note**
>
> Remember to include the LDAP connection parameters such as host name, port, bindDN, and bindPassword.

2. If the LDAP changelog backend at the previous location is enabled:

   1. Stop the server.
   2. Copy the database files to the new LDAP changelog location.
   3. Restart the server.

> **Note**
>
> The server attempts to use whatever it finds in the configured location when it starts. If there is nothing there, it creates an empty database.

Viewing the LDAP changelog parameters in the Root DSE

The Root DSE is a special entry that holds operational information about the server.

About this task

The entry provides information about the LDAP controls, extended operations, and Simple Authentication and Security Layer (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 changelog with data in the LDAP changelog of other servers in the replication topology. The order of the individual changes in the LDAP changelog can vary between servers based on the order in which they were received from a replica.

Steps

- Use `ldapsearch` to view the Root DSE.

  Example:

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

Viewing the LDAP changelog, change sequence numbers, and monitoring information

View changelog entries using `ldapsearch`.

All records in the changelog are immediate children of the `cn=changelog` entry and are named with the `changeNumber` attribute. Changes are represented in the form documented in the `draft-good-ldap-changelog` specification with the `targetDN` attribute providing the distinguished name (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. View the changes by decoding the encoded value using the `base64 decode` utility. The UnboundID LDAP SDK for Java also provides support for parsing changelog entries.

Viewing the LDAP changelog using ldapsearch

Steps

1. By default, only users with the `bypass-acl` or `bypass-read-acl` privilege can access changelog entries. To grant control permission to allow other users to see changelog entries, use a global ACI like the following:

   Example:

   ```
   $ bin/dsconfig set-access-control-handler-prop
   --add 'global-aci:(targetattr="*||+")(target="ldap:///cn=changelog")(version 3.0;
   acl "Access to the changelog backend for the admin account";
   allow (read,search,compare) userdn="ldap:///uid=admin,dc=example,dc=com";)'
   ```

2. Use `ldapsearch` to view the changelog.

   Example:

   ```
   $ bin/ldapsearch --hostname ds.example.com --port 636 --useSSL
   --bindDN "uid=admin,dc=example,dc=com" --bindPasswordFile admin-password.txt
   --baseDN cn=changelog --dontWrap "(objectclass=*)"
   ```

   Result:

```
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:: cmVwbGFjZTogZHMtc3luY1zdGF0ZQpkcy1zeW5jLXN0YXRlOiAwMDAwMDExQ0ZGMzM0Q
zYwNDA5MzAwMDAwMDAyCgo=
changenumber: 2
```

Viewing the LDAP change sequence numbers

About this task

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.

Steps

- Make sure the `uid=admin` account has the necessary access rights to the `cn=changelog` backend.

  Example:

  ```
  $ bin/ldapsearch --hostname ds.example.com --port 636 --useSSL
  --bindDN "uid=admin,dc=example,dc=com" --bindPasswordFile admin-password.txt
  --baseDN cn=changelog --dontWrap "(objectclass=*)" "+"
  ```

  Result:

```
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 accessed over LDAP, JConsole, the admin console, or SNMP. Make sure the account you're using to request the monitor information has the necessary access rights to the data under `cn=monitor`. You might need to add a global ACI to grant the appropriate users permission to access monitor data.

Steps

- Use `ldapsearch` to view the changelog monitor entry.

  Example:

  ```
  $ bin/ldapsearch --hostname ds.example.com --port 636 --useSSL
  --bindDN "uid=admin,dc=example,dc=com" --bindPasswordFile admin-password.txt
  --baseDN cn=changelog,cn=monitor "(objectclass=*)"
  ```

  Result:

  ```
  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 PingDirectory 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 PingDirectory 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 or not the changelog entry includes changes to a particular attribute or set of attributes. Using changelog indexing, client applications can dramatically speed up throughput when targeting the specific attributes.

You 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 PingDirectory server supports this feature, 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.

   Example:

   Index different attributes by adding specific properties. In the following example, the 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 are 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. To add another attribute to index, use the `dsconfig` `--add` option, which adds the attribute to an existing configuration setting.

   Example:

   ```
   $ 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.

  Example:

  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

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.

About this task

The `include-virtual-attributes` property controls the virtual attribute information to include in changelog entries, and this property might 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` values are selected by default.

To enable virtual attribute change tracking in the LDAP Changelog:

Steps

- Use `dsconfig`.

Example:

> **Note**
>
> The following command enables the LDAP changelog and sets `include-virtual-attributes` to `add-attributes`, which indicates that virtual attributes be included in the set of attributes listed for an add operation. The `delete-entry-attributes` value indicates that virtual attributes should be included in the set of deleted entry attributes listed for a delete operation. The `before-and-after-values` value 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` value 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 the schema

This section is a summary of the supported schema components on the PingDirectory server and procedures to extend the schema with new element definitions.

About the schema

A schema is the set of PingDirectory 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 PingDirectory server objects.

The PingDirectory server includes a default set of read-only schema files that define the core properties for the server. The admin console provides a Schema Editor to view existing schema definitions and add new custom schema elements to your DIT.

> **Note**
>
> Any attempt to alter a schema element defined in a read-only file or add a new schema element to a read-only file results in an `Unwilling to Perform` result.

About the schema editor

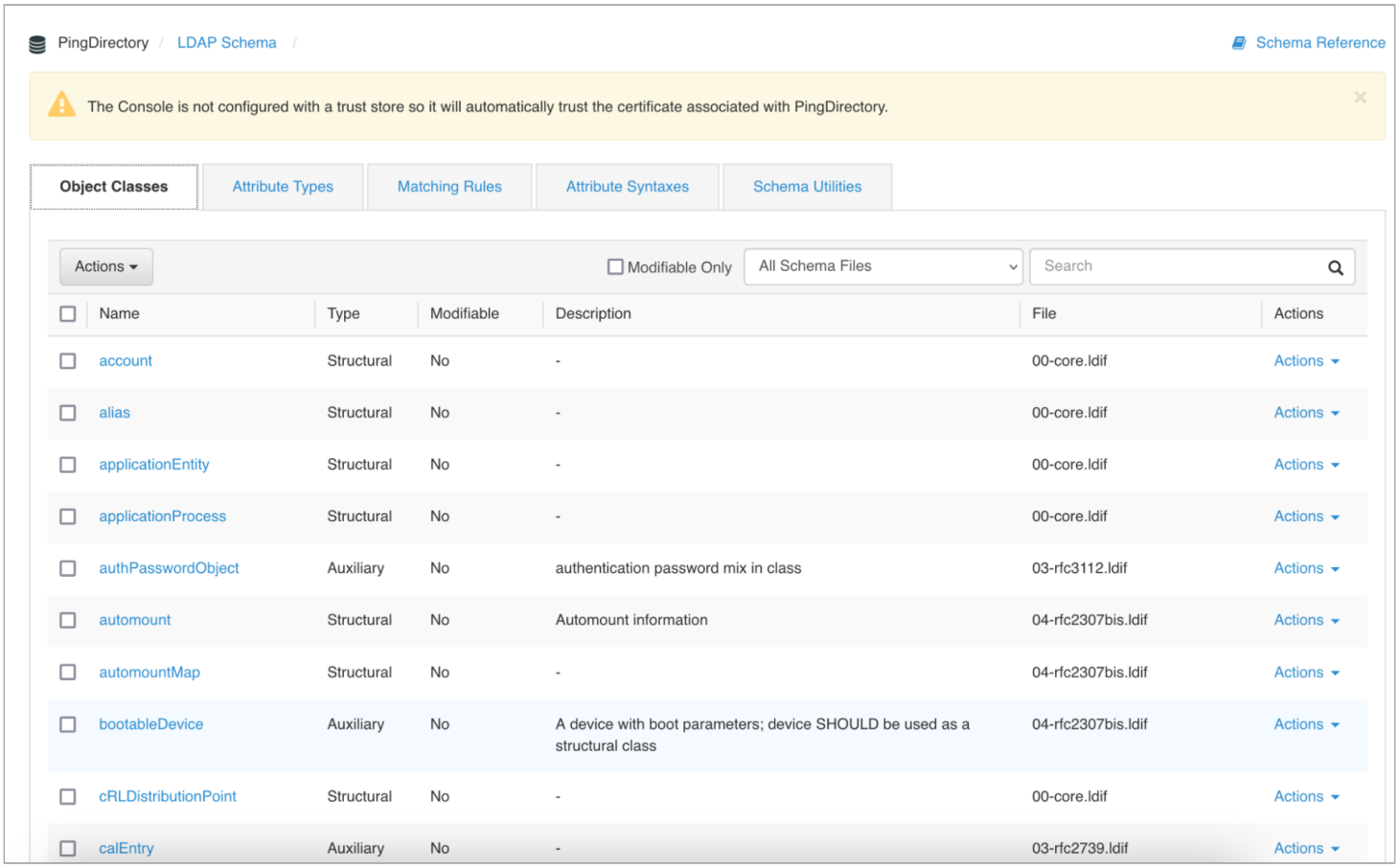
The admin console provides a user-friendly graphical editor with tabs to manage any existing schema components related to the directory information tree (DIT).

The list of existing schema components that can be managed includes:

- Object classes
- Attributes
- Matching rules
- Attribute syntaxes
- Schema utilities

The following list describes the tabs of the Schema Editor page:

- 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 supports schema file imports by first checking for proper syntax compliance and generating any error message if the definitions do not meet specification.



Example schema editor page

The schema editor provides two views for each definition:

- The Properties View that breaks down the schema definition by its properties and shows any inheritance relationships among the attributes.
- The LDIF View that shows the equivalent schema definition in ASN.1 format, which includes the proper text spacing required for each schema element.

Default PingDirectory server schema files

The PingDirectory server stores its schema as a set of LDIF files for the server instance in the `<server-root>/config/schema` directory. The PingDirectory server reads the schema files in alphanumeric order at startup in the following order:

1. The `00-core.ldif` file
2. The `01-pwpolicy.ldif` file
3. Remaining files

You should name custom schema files so that they are loaded in last. For example, custom schema elements could be saved in a file labeled `99-user.ldif` that loads after the default schema files are read at startup.

The PingDirectory server then uses the schema definitions to determine any violations that might occur during `add`, `modify`, or `import` requests. Clients applications check the schema (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 PingDirectory server's core functions
`01-pwpolicy.ldif`	Governs password policies
`02-config.ldif`	Governs the PingDirectory server's configuration
`03-changelog.ldif`	Governs the PingDirectory 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 Service Oriented Architecture (SOA) components
`04-rfc2307bis.ldif`	Governs mapping entities from TCP/IP and UNIX into X.500 entries

Extending the PingDirectory server schema

The PingDirectory server stores its schema as LDIF files in the `<server-root>/config/schema` directory.

At startup, the PingDirectory server reads the schema files 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 PingDirectory 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 PingDirectory server reads the file after the core schema files are read at startup. For example, you can name the file `99-myschema.ldif`.

  > **Note**
  >
  > Learn more about the requirements for naming each file in General tips on extending the schema.

- 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 PingDirectory server writes the new schema changes to a file such as `99-user.ldif` in the `<server-root>/config/schema` directory. However, this method can be cumbersome because schema definitions require strict adherence to text spacing and white space characters.

General tips on extending the schema

Consider the following points when extending the schema:

- Never modify the default schema files because doing so could damage the PingDirectory server's processing capabilities.
- Define all attributes before they can be used in an object class. If you are using the Schema Editor to add new schema elements, use the `Quick Add Attributes` option when defining new object classes.
- Define the parent object class before creating object classes that inherit from the parent.
- You must name custom schema files according to the following syntax:

  1. Begin with exactly two digits.
  2. Follow the two digits with a non-digit character.
  3. Follow the non-digit character with a zero or more characters.
  4. End in `.ldif`.

  > **Note**
  >
  > The two digits don't need to be followed by a dash ("-").
  > Any files that do not meet this criteria are ignored and either a `NOTICE` or `SEVERE_WARNING` message is logged.

  Any file in the `<server-root>/config/schema` directory with a name that starts with "." or ends with a tilde ('~'), `.swp`, or `.tmp` generates a `NOTICE` message indicating that temporary files will be ignored. Any other file that does not meet the naming criteria generates a `SEVERE_WARNING` message indicating that it will be ignored.

- Define custom attributes and object classes in one file. Typically, this file is the `99-user.ldif`. You can specify a different file name that the PingDirectory server writes to using the `X-SCHEMA-FILE` element and the file name in the definition, as in the following 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' )
```

- In the white space characters in the schema definitions, `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 because continuation lines must begin with a space character.

  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 guarantee correct formatting. 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 are replicated to other replication servers to their respective Schema backend. The additions are written to the file specified by the `X-SCHEMA-FILE` extension or written to `99-user.ldif` if no file is specified.

About 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, that 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 used for administrative or server-related purposes.

  > **Tip**
  >
  > To specify the purpose of the attribute, use the `USAGE` element.

- Attributes are multi-valued by default. Multi-valued means that attributes can contain more than one value within an entry.

> **Tip**
>
> If the attribute should contain at most one value within an entry, include the `SINGLE-VALUE` element.

- 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 that 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 don't 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. For more information, see Extending the PingDirectory server schema.

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 following table details the standard elements in schema definition.

Basic Properties of Attributes

Attributes	Description
`Name`	The globally unique name
`Description`	An optional definition that describes the attribute and its contents The LDIF equivalent is `DESC`.
`OID`	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`	The attribute syntax used For example, the `userPassword` attribute uses the User Password Syntax whereas the `authPassword` attribute uses the Authentication Password Syntax.
`Parent`	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

Attributes	Description
`Value Restrictions`	Specifies any restriction on the value of the attribute

The Additional Properties table details auxiliary information associated with the attribute.

Attributes	Description
`Aliases`	Any shortform alias names, if any You can 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`	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`	The intended use of the attribute The choices are: • `userApplications` • `directoryOperation` • `distributedOperation` • `dSAOperation`
`User-Modifiable`	Specifies if the attribute can be modified by an authorized user
`Obsolete`	Specifies if the schema definition is obsolete or not
`Matching Rules`	Specifies the associated matching rules for the attribute

Viewing attributes

The Schema Editor displays all of the attribute types on your PingDirectory server instance.

It shows the basic properties that are required elements in addition to the extra properties that are allowed within the attribute definition.

Viewing attribute types using the Schema Editor

Before you begin

Ensure that the PingDirectory server instance is running.

About this task

To view attribute types using the Schema Editor:

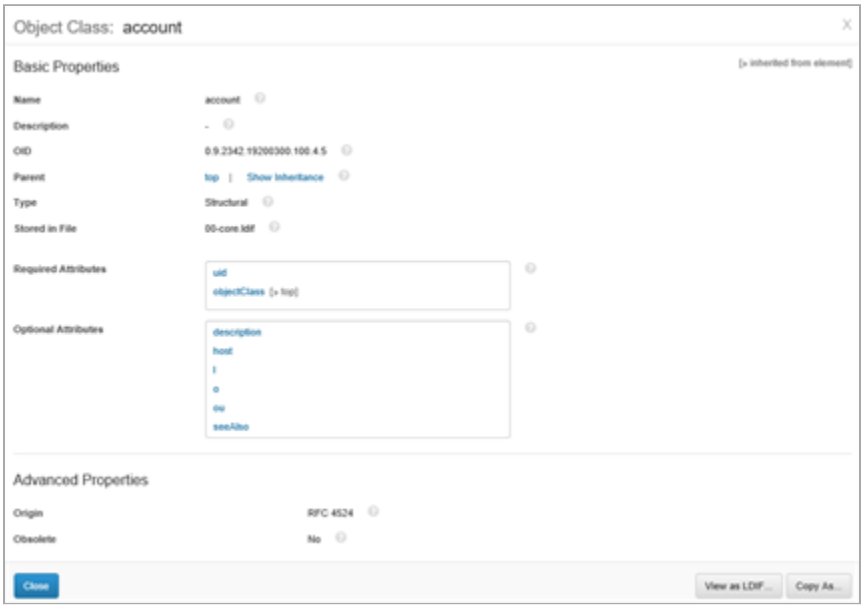
Steps

1. Start the admin console.
2. In the main menu, select Schema.
3. In the Schema Editor, click the Attribute Types tab.
4. Click a specific attribute to view its definition.

   In the Object Class window, view the attribute properties.

   Example:

   To follow this example, click the `account` attribute.



5. To see the equivalent attribute definition in ASN.1 format, click the View as LDIF button.

Viewing attribute types over LDAP

Steps

- To view an operational attribute, run `ldapsearch`.

  Example:

  This example uses `ldapsearch` to view the multi-valued operational attribute `attributeTypes`, which publishes the definitions on the PingDirectory server.

> **Note**
>
> The attribute is stored in the subschema subentry.

```
$ bin/ldapsearch --baseDN cn=schema --searchScope base \
  "(objectclass=*)" attributeTypes
```

Viewing a specific attribute type over LDAP

Steps

- To search for a specific attribute, run `ldapsearch` with the `--dontWrap` option and use the `grep` command.

  Example:

```
$ bin/ldapsearch --baseDN cn=schema \
  --searchScope base --dontWrap "(objectclass=*)" \
  attributeTypes | grep 'personalTitle'
```

Creating a new attribute over LDAP

The following sections demonstrate how to add a schema element over LDAP.

You can create your own schema file or type the schema from the command line.

> **Note**
>
> Make sure you're aware of text spacing and ASN.1 formatting.

Adding a new attribute to the schema over LDAP

Steps

1. In a text editor, create an LDIF file with the new attribute definition.

   Example:

   In this example, the LDIF file is named `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. To add the attribute, run `ldapmodify`.

   Example:

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

3. To verify the addition, display the attribute using `ldapsearch`.

```
$ bin/ldapsearch --baseDN cn=schema --searchScope base \
  --dontwrap "(objectclass=*)" attributeTypes | grep 'contractorStatus'
```

4. To view the custom schema file, go to `<server-root>/config/schema/99-user.ldif`.

   Result:

   For this example, you see the following details:

```
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' )
```

Adding constraints to attribute types

About this task

The PingDirectory server provides attribute type extensions that constrain the values for the associated attribute using the `DirectoryString` attribute syntax.

To constrain the values for an attribute:

Steps

- Use the `DirectoryString` attribute syntax.

  Example:

  The following example schema definition includes two `attributeType` definitions for `myAttr1` and `myAttr2`:

  - The first definition constrains the values for the attribute `myAttr1` to `'foo'`, `'bar'`, and `'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 PingDirectory server uses the schema to do the following:

- Define the entries.
- Specify the position of the entries in a DIT.
- Control the operations 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 might 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.

> **Note**
>
> The PingDirectory server also has a special entry called the subschema subentry, which provides information about the available schema elements on the server.

Object class types

Based on RFC 4512, object classes can be a combination of three different types:

Abstract object classes

Used as the base object class from which structural or auxiliary classes inherit its properties.

This inheritance is a one-way relationship because 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 directory information tree (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

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

You can specify new object classes with existing schema components that don't require additional server code extensions for their implementation. To create new object classes, use the Schema Editor, which manages schema in the `<server-root>/config/schema` directory. Learn more in Extending the PingDirectory server schema.

RFC 4512, section 4.1.1, defines the following object class definition:

```
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 PingDirectory server and aren't 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, 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 table displays the standard elements in schema definition.

Attributes	Description
`Name`	The globally unique name.
`Description`	An optional definition that describes the object class and its contents. The analogous LDIF equivalent is `DESC`.
`OID`	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`	The schema definition's hierarchical parent or superior object class, if any. An object class can have one parent.
`Type`	The type of schema definition, which can be abstract, structural, or auxiliary. The analogous LDIF equivalent is `ABSTRACT`, `STRUCTURAL`, or `AUX`.
`Required Attributes`	Specifies required attributes with the object class. The analogous LDIF equivalent is `MUST`. **Tip** The Schema Editor marks any inherited attributes from another object class. Double-click an attribute value to take you to the Properties View for that particular attribute.
`Optional Attributes`	Specifies optional attributes that can be used with the object class. The analogous LDIF equivalent is `MAY`. **Tip** The Schema Editor marks any inherited attributes from another object class. Double-click an attribute value to take you to the properties view for that particular attribute.

The Additional Properties table provides auxiliary information associated with the object class.

Attributes	Description
`Aliases`	Specifies short-form alias names, if any. You could have any number of short-form 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`	The origin of the schema definition. Typically, it could refer to a specific RFC or company.

Attributes	Description
`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

About this task

To view the object classes on your PingDirectory server, use any of the following:

- The admin console Schema Editor
- LDAP with the `ldapsearch` tool
- A third-party tool

> **Note**
>
> The Schema Editor displays all of the object classes on the PingDirectory 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

- To view an operational attribute, run `ldapsearch`.

  Example:

  This example uses `ldapsearch` to view the multi-valued operational attribute `objectClasses`, which publishes the object class definitions on the PingDirectory server.

  > **Note**
  >
  > The attribute is stored in the subschema subentry.

  ```
  $ bin/ldapsearch --baseDN cn=schema --searchScope base \
  --dontWrap "(objectclass=*)" objectClasses
  ```

  Result:

  ```
  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

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.

Before you begin

Define the attribute you want to add to the custom schema file.

About this task

The following example procedure adds a predefined attribute, `contractorAddress`, to the custom schema file, then adds it to the `contractor` object class.

Steps

1. Create and save an LDIF file with the content in the following example.

   Example:

   In this example, the file is named `contractorAddress-attr.ldif`.

   ```
   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. To add the attribute you defined, run `ldapmodify`.

   Example:

   In this example, the `contractorAddress` attribute is being added.

   ```
   $ bin/ldapmodify --filename contractorAddress-attr.ldif
   ```

3. To modify the contractor object class to allow this attribute, create an LDIF file.

   When doing this, you are re-adding the updated `objectClass`. The PingDirectory server handles the proper replacement of the existing object class with the new one.

   > **Note**
   >
   > Ensure that the lines are not wrapped, the `objectClasses` line should be one continuous line.

Example:

In this example, the file is named `contractor-oc.ldif`.

```
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. To update the `objectClass`, run `ldapmodify`.

   Example:

   ```
   $ bin/ldapmodify --filename contractor-oc.ldif
   ```

   Result:

   These schema changes are replicated to all servers in the replication topology.

5. To verify the change, view the `config/schema/98-custom-schema.ldif` file on the other servers in the replication topology to ensure that the changes are present.

6. (Optional) To add an index for this attribute, run `dsconfig` with the `create-local-db-index` option.

   > **Note**
   >
   > You must do this on each server in your topology unless you have server configuration groups set up. For more information, see Configuring server groups.

   Example:

   ```
   $ bin/dsconfig create-local-db-index --backend-name userRoot \
     --index-name contractorAddress --set index-type:equality
   ```

7. Rebuild the index online.

   > **Note**
   >
   > This doesn't affect other indexes or entries because there is no existing data for this attribute on any entry.

   Example:

   ```
   $ bin/rebuild-index --baseDN dc=example,dc=com --index contractorAddress
   ```

Creating a new object class using the schema editor

Before you begin

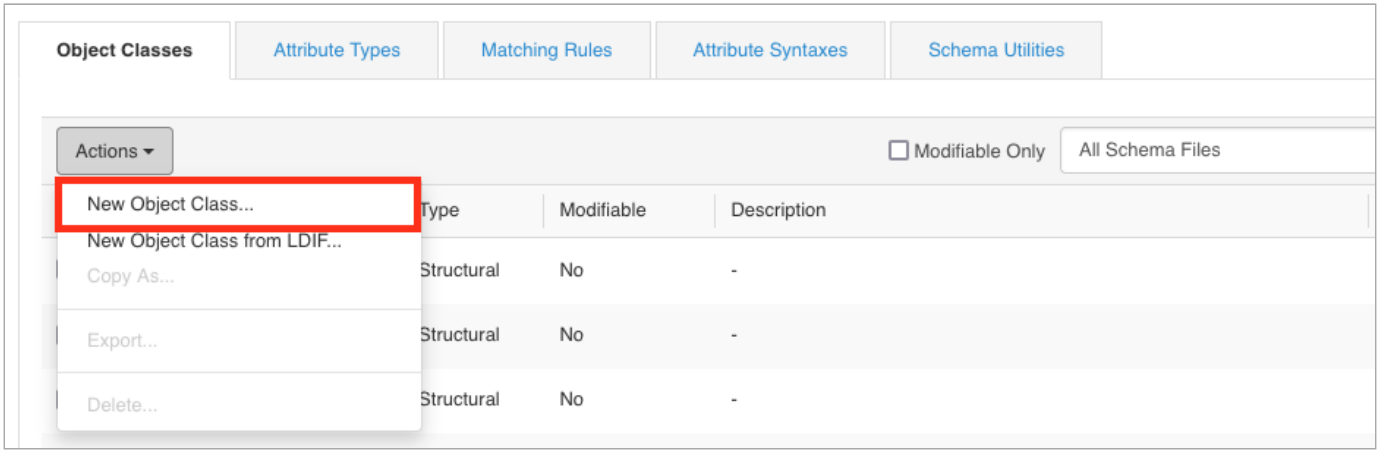
Make sure that any attributes that are part of the new object class are defined before creating the object class.

About this task

Creating a new object class is similar to creating a new attribute. To create a new object class:

Steps

1. Start the admin console.
2. In the top-level navigation menu, select LDAP Schema.
3. Click the Object Classes tab, and then in the Actions list, select New Object Class.



4. Enter the properties for the new object class.
5. 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.

> **Note**
>
> All custom attributes appear at the bottom of the list in the **Attributes** box.

Extending the schema using a custom schema file

To add new attributes and object classes to your PingDirectory server schema, create a custom schema file.

Steps

1. Create an LDIF file with the new attribute extensions using a text editor.

   Example:

```
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 LDIF file you created in step 1, add a new object class definition after the attribute types.

   Example:

   This example creates an auxiliary object class, `contractor`, that alone cannot be used as an entry.

   The object class is 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 and place it in the `<server-root>/config/schema` directory.

   In this example, the file is saved as `99-auxobjclass.ldif`.

4. Load the schema extensions into the PingDirectory server. You have four options:

   Choose from:

   - Create a task that loads the new extensions into the schema.

     The following example creates a task with the ID `add-schema-99-auxobjclass` and adds it using `ldapmodify`.

```
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
```

Note

When using this method, you don't need to restart the server.

- Import the schema file using the admin console Schema Editor.

Note

When using this method, you don't need to restart the server.

1. Place the `99-auxobjclass.ldif` file in the `<server-root>/config/schema` directory.
2. Restart PingDirectory server.

Note

The schema file is read at startup.

- Add the schema file using `load-ldap-schema-file`.

```
$ bin/load-ldap-schema-file --schemaFile config/schema 99-auxobjclass.ldif
```

Note

When using this method, you don't need to restart the server.

5. Add the new object class and attribute to an existing user entry.

Example:

```
$ bin/ldapmodify
dn: uid=user.9,ou=People,dc=example,dc=com
changetype: modify
add: objectClass
objectClass: contractor
-
add: contractorStatus
contractorStatus: TRUE
```

6. To verify the addition, run `ldapsearch` to display the attribute.

Example:

```
$ bin/ldapsearch --baseDN dc=example,dc=com "(uid=user.9)" contractorStatus
```

Result:

```
dn: uid=user.9,ou=People,dc=example,dc=com
contractorStatus: TRUE
```

About managing matching rules

Matching rules determine how clients and servers compare attribute values during LDAP requests or operations.

Matching rules are also used in evaluating search filter elements, including distinguished names (DNs) and attributes. They are defined for each attribute based on the following properties:

- `EQUALITY`: two attributes are equal based on case, exact match, and so forth
- `SUBSTR`: assertion value is a substring of an attribute
- `ORDERING`: greater than or equal, less than or equal, and so forth

> **Note**
>
> PingDirectory server supports an `APPROXIMATE` matching rule that compares similar attributes based on fuzzy logic. Attributes that are similar or sound alike 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 a Ping Identity support representative for assistance.

The following formal specification for attribute types is provided in RFC 4512, section 4.1.3.

```
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

PingDirectory server provides a large set of matching rules that support a variety of applications.

The default matching rules available for the PingDirectory server are listed in the table below for each matching rule type:

- Equality matches
- Substring matches
- Ordering matches
- 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 (IA5) 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 IA5 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

Matching Rule/OID	Attribute Syntax/OID	Description
`ds-mr-double-metaphone-approx/ 1.3.6.1.4.1.30221.1.4.1`	`Directory String/ 1.3.6.1.4.1.1466.115.121.1.15`	Syntax based on the phonetic Double Metaphone algorithm for approximate matching
`ds-mr-user-password-exact/ 1.3.6.1.4.1.30221.1.4.2`	`User Password Syntax/ 1.3.6.1.4.1.30221.1.3.1`	User password exact matching rule
`ds-mr-user-password-equality/ 1.3.6.1.4.1.30221.1.4.3`	`User Password Syntax/ 1.3.6.1.4.1.30221.1.3.1`	User password equality matching rule
`historicalCsnOrderingMatch/ 1.3.6.1.4.1.30221.1.4.4`	`1.3.6.1.4.1.30221.1.3.5`	Compares the collation order of a historical change sequence number with a historical Change Sequence Number (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 distinguished name (DN) syntax Spaces around commas and semicolons are ignored. Spaces around plus and equal signs around relative distinguished name (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

Matching Rule/OID	Attribute Syntax/OID	Description
`caseIgnoreOrderingMatch/ 2.5.13.3`	`Directory String/ 1.3.6.1.4.1.1466.115.121.1.15`	Compares the collation order of the asserted string with an attribute value of Directory String syntax Case, leading, and trailing spaces are ignored. Multiple spaces are treated as a single space. RFC 4517
`caseIgnoreSubstringsMatch/ 2.5.13.4`	`Substring Assertion/ 1.3.6.1.4.1.1466.115.121.1.58`	Compares an asserted substring value with an attribute value of Directory String syntax Case, leading, and trailing spaces are ignored. Multiple spaces are treated as a single space. RFC 4517
`caseExactMatch/ 2.5.13.5`	`Directory String/ 1.3.6.1.4.1.1466.115.121.1.15`	Compares an asserted value with an attribute value of Directory String syntax RFC 4517
`caseExactOrderingMatch/ 2.5.13.6`	`Directory String 1.3.6.1.4.1.1466.115.121.1.15`	Compares the collation order of the asserted string with an attribute value of Directory String syntax RFC 3698
`caseExactSubstringsMatch/ 2.5.13.7`	`Substring Assertion/ 1.3.6.1.4.1.1466.115.121.1.58`	Compares an asserted substring with an attribute value of Directory String syntax RFC 3698
`numericStringMatch/ 2.5.13.8`	`Numeric String/ 1.3.6.1.4.1.1466.115.121.1.36`	Compares an asserted value with an attribute value of Numeric String syntax. Spaces are ignored when performing these comparisons RFC 4517.
`numericStringOrderingMatch/ 2.5.13.9`	`Numeric String/ 1.3.6.1.4.1.1466.115.121.1.36`	Compares the collation order of the asserted string with an attribute value of Numeric String syntax Spaces are ignored when performing these comparisons. RFC 4517

Matching Rule/OID	Attribute Syntax/OID	Description
`numericStringSubstringsMatch/ 2.5.13.10`	`Substring Assertion/ 1.3.6.1.4.1.1466.115.121.1.58`	Compares an asserted substring with an attribute value of Numeric String syntax Spaces are ignored when performing these comparisons. RFC 4517
`caseIgnoreListMatch/ 2.5.13.11`	`Postal Address/ 1.3.6.1.4.1.1466.115.121.1.41`	Compares an asserted value with an attribute value, which is a sequence of Directory Strings Case, leading, and trailing spaces are ignored. Multiple spaces are treated as a single space. RFC 4517
`caseIgnoreListSubstringsMatch/ 2.5.13.12`	`Substring Assertion/ 1.3.6.1.4.1.1466.115.121.1.58`	Compares the asserted substring with an attribute value, which is a sequence of Directory Strings Case, leading, and trailing spaces are ignored. Multiple spaces are treated as a single space. RFC 3698
`booleanMatch/ 2.5.13.13`	`Boolean/ 1.3.6.1.4.1.1466.115.121.1.7`	Compares an asserted boolean value with an attribute value of BOOLEAN syntax Returns true if the values are both `TRUE` or both `FALSE`. RFC 3698
`integerMatch/ 2.5.13.14`	`Integer/ 1.3.6.1.4.1.1466.115.121.1.27`	Compares an asserted value with an attribute value of INTEGER syntax RFC 4517
`integerOrderingMatch/ 2.5.13.15`	`Integer/ 1.3.6.1.4.1.1466.115.121.1.27`	Compares the collation order of the asserted integer with an attribute value of Integer syntax Returns true if the attribute value is less than the asserted value. RFC 3698
`bitStringMatch/ 2.5.13.16`	`Bit String/ 1.3.6.1.4.1.1466.115.121.1.6`	Compares an asserted Bit String value with an attribute value of Bit String syntax RFC 4517

Matching Rule/OID	Attribute Syntax/OID	Description
`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
`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

Matching Rule/OID	Attribute Syntax/OID	Description
`generalizedTimeOrderingMatch/ 2.5.13.28`	`Generalized Time 1.3.6.1.4.1.1466.115.121.1.24`	Compares the collation order of the asserted string with an attribute value of Generalized Time String syntax and case is ignored RFC 4517
`integerFirstComponentMatch/ 2.5.13.29`	`Integer/ 1.3.6.1.4.1.1466.115.121.1.27`	Equality matching rules for subschema attributes between an Integer syntax and the value syntax RFC 4517
`objectIdentifierFirstComponentMatch/ 2.5.13.30`	`OID/ 1.3.6.1.4.1.1466.115.121.1.38`	Equality matching rules for subschema attributes between an OID syntax and the value syntax RFC 4517
`directoryStringFirstComponentMatch/ 2.5.13.31`	`Directory String/ 1.3.6.1.4.1.1466.115.121.1.15`	Compares an asserted Directory String value with an attribute value of type `SEQUENCE` whose first component is mandatory and of type Directory String Returns true if the attribute value has a first component whose value matches the asserted Directory String using the rules of `caseIgnoreMatch`. RFC 3698.
`wordMatch/ 2.5.13.32`	`Directory String/ 1.3.6.1.4.1.1466.115.121.1.15`	Compares an asserted word with any word in the attribute value for equality RFC 3698
`keywordMatch/ 2.5.13.33`	`Directory String/ 1.3.6.1.4.1.1466.115.121.1.15`	Compares an asserted value with any keyword in the attribute value for equality RFC 3698

Basic matching rule properties

The following table describes the standard elements in a matching rule schema definition.

Attributes	Description
`Name`	Specifies the descriptive and unique name of the element

Attributes	Description
`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 matching rule: • `Equality` • `Ordering` • `Substring` • `Approximate`
`Syntax`	Specifies the matching rule syntax
`Used by Attributes`	Specifies any attributes that use the corresponding matching rule

Viewing matching rules

To view the matching rules on your PingDirectory server, use the Schema Editor, LDAP with `ldapsearch`, or a third-party tool.

About this task

To view matching rules over LDAP:

Steps

- To view the `matchingRules` attribute, run `ldapsearch`.

  Example:

  The following example uses `ldapsearch` to view a multi-valued operational attribute, `matchingRules`, which publishes the definitions on the PingDirectory server.

  > **Note**
  >
  > The attribute is stored in the subschema subentry.

```
$ bin/ldapsearch --baseDN cn=schema --searchScope base \
  "(objectclass=*)" matchingRules
```

About 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 Ping Identity support for assistance.

The following formal specification for attribute types is provided in RFC 4512, section 4.1.5.

```
SyntaxDescription = "(" wsp
numericoid                    ; Object identifier
[ sp "DESC" sp qdstring ]     ; Description
extensions wsp ")"            ; Extensions followed by a white space and ")"
```

Default attribute syntaxes

PingDirectory server supports a large set of Attribute Syntax rules for applications.

The default Attribute Syntax rules available for the server are listed in the following table.

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`

LDAP Syntax	OID	Description
`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`, and so forth **Note** The current implementation for Country String: • Only verifies that values are two printable characters, where the set of printable characters is defined in RFC 2252. • Does not check that the value is a valid ISO 3166 country code.
`DN`	`1.3.6.1.4.1.1466.115.121.1.12`	Distinguished name of an entry as defined in RFC 4514
`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 RFC 4517 For example, `videotext $ telephone`
`Directory String`	`1.3.6.1.4.1.1466.115.121.1.15`	String of one or more characters from the Universal Character Set (UCS) using UCS Transformation Format 8 (UTF-8) encoding of the string
`DIT Content Rule Description`	`1.3.6.1.4.1.1466.115.121.1.16`	`DITContentRuleDescription` as defined in RFC 4517
`DIT Structure Rule Description`	`1.3.6.1.4.1.1466.115.121.1.17`	`DITStructureRuleDesciption` as defined in RFC 4517
`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 RFC 4517 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 RFC 4517
`Fax`	`1.3.6.1.4.1.1466.115.121.1.23`	Image generated using Group 3 fax process as defined in RFC 4517

LDAP Syntax	OID	Description
`Generalized Time`	`1.3.6.1.4.1.1466.115.121.1.24`	String representing data and time as defined in RFC 4517 `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 RFC 4517
`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 RFC 4517
`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 RFC 4517
`JPEG`	`1.3.6.1.4.1.1466.115.121.1.28`	Image in JPEG File Interchange Format (JFIF) as defined in RFC 4517
`Matching Rule Description`	`1.3.6.1.4.1.1466.115.121.1.30`	`MatchingRuleDescription` as defined in RFC 4512
`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 RFC 4511
`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 distinguished names (DNs) as defined in RFC 4517 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 RFC 4512
`Numeric String`	`1.3.6.1.4.1.1466.115.121.1.36`	Sequence of one or more numerals and spaces as defined in RFC 4517 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 RFC 4517

LDAP Syntax	OID	Description
`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$USA`
`Protocol Information`	`1.3.6.1.4.1.1466.115.121.1.42`	Undefined
`Presentation Address`	`1.3.6.1.4.1.1466.115.121.1.43`	String encoded OSI presentation address as defined in RFC 1278 For example, `TELEX+00728722+RFC-1006+03+10.0.0.6`
`Printable String`	`1.3.6.1.4.1.1466.115.121.1.44`	String of one or more printable ASCII alphabetic, numeric, and punctuation characters as defined in RFC 4517
`RFC3672 Subtree Specification`	`1.3.6.1.4.1.1466.115.121.1.45`	Syntax based on subtree specification as defined as RFC 3672
`Supported Algorithm`	`1.3.6.1.4.1.1466.115.121.1.49`	Octet string based on the LDAP-encoding for a supported algorithm value that results from the BER encoding of a `SupportedAlgorithm` ASN.1 value
`Telephone Number`	`1.3.6.1.4.1.1466.115.121.1.50`	String of printable international telephone number representations in E.123 format as defined in RFC 4517 For example, `+1 512 904 5525`
`Teletex Terminal Identifier`	`1.3.6.1.4.1.1466.115.121.1.51`	Identifier and telex terminal as defined in RFC 4517
`Telex Number`	`1.3.6.1.4.1.1466.115.121.1.52`	String representing the telex number, country code, and answerback code as defined in RFC 4517 For example, `812374, ch, ehhg`
`UTC Time`	`1.3.6.1.4.1.1466.115.121.1.53`	Character string representing the data and time in UTC Time format as defined as RFC 4517: `YYMMDDHHMM[SS][(+|-HHMM)|Z]`, where `Z` is the coordinated universal time. For example, `0903051035Z, 0903051035-0500`
`LDAP Syntax Description`	`1.3.6.1.4.1.1466.115.121.1.54`	`SyntaxDescription` as defined inhttps://tools.ietf.org/html/rfc4512[RFC 4512]
`Substring Assertion`	`1.3.6.1.4.1.1466.115.121.1.58`	Syntax for assertion values in an extensible match as defined in RFC 4517
`Authentication Password Syntax`	`1.3.6.1.4.1.4203.1.1.2`	Encoded password storage syntax as defined in RFC 3112 For example, the syntax specifies the storage scheme in brackets: `<storage-scheme>$<auth component>$<auth value>` For example, `SSHA$xdEZWRqgyJk=$egDEFDXvdeeEnXUEIDPnd39dkpe=`

LDAP Syntax	OID	Description
`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: `{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 following table provides a description of the standard elements in an attribute syntax.

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

To view the attribute syntaxes on your PingDirectory server, use the Schema Editor, LDAP with `ldapsearch`, or a third-party tool.

About this task

To view attribute syntaxes over LDAP:

Steps

- To view the PingDirectory server's published list of attribute syntaxes, run `ldapsearch` using the `ldapSyntaxes` attribute.

  Example:

The following example uses `ldapsearch` to view the PingDirectory server's published list of attribute syntaxes using the multi-valued operational attribute, `ldapSyntaxes`, which publishes the definitions on the server.

> **Note**
>
> 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 where you can import new schema elements from a file and check schema compliance.

About this task

If you are importing a schema file, the system automatically checks for compliance before the import. If the definition doesn't meet schema compliance, the system displays an error message. However, you should check if your file is compliant with your schema before importing it.

To check schema compliance using the schema editor:

Steps

1. Start the admin console.
2. In the top-level navigation menu, click LDAP Schema.
3. In the schema editor, click the Schema Utilities tab.
4. Add your schema definition using one of two methods:

   Choose from:

   - To have the system check an LDIF file, click Import Schema Elements and select a file to upload.
   - Copy and paste a new schema definition into the field.
5. Click Validate Entries.

   Result:

   If there is a problem with your definition, you see an error message.

Modifying a schema definition

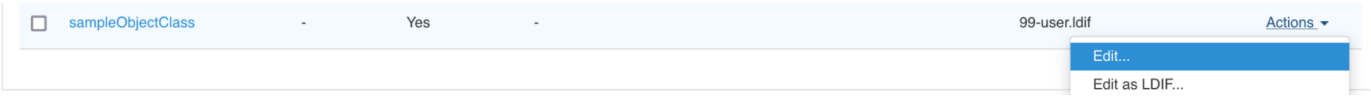
About this task

You can only edit schema definitions in the PingDirectory server that are read-write. The 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 admin console.
2. In the top-level navigation menu, click Schema.
3. Click the Object Classes tab.
4. Select the object class that you want to modify, and then click Actions → Edit.



5. Make your changes and click OK.

Deleting a schema definition

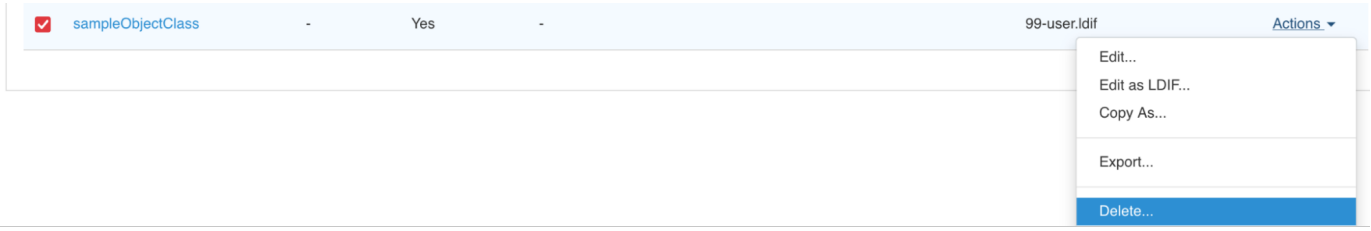
About this task

You can only delete schema definitions in the PingDirectory server that are read-write. In general, you can only remove schema definitions in the Custom folder of the schema editor. Ensure that the schema element you are deleting is not currently in use.

To delete a schema definition:

Steps

1. Start the admin console.
2. In the top-level navigation menu, click Schema.
3. Click the Object Classes tab.
4. Select the object class that you want to remove, and then in the Actions list, select Delete.



Result:

The Confirmation dialog box opens.

5. To delete the schema element, click Yes.

Managing schema checking

The PingDirectory server provides full support for parsing all schema elements and provides access to all of its components.

By default, the PingDirectory 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 generate an error message to standard output.

Viewing the schema checking properties

Steps

- To view the schema checking property, run `dsconfig` with the `get-global-configuration-prop` option.

  Example:

  ```
  $ bin/dsconfig get-global-configuration-prop \
    --property check-schema
  ```

Disabling schema checking

Although you can use the `dsconfig` tool to disable the schema checking, it's not recommended.

About this task

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 is generated if you attempt to disable schema checking using `dsconfig` in interactive or non-interactive mode. The alert provides alternatives to disabling schema checking.

Steps

1. To disable the `check-schema` property, run `dsconfig` with the `set-global-configuration-prop` option.

   Example:

   ```
   $ bin/dsconfig --no-prompt set-global-configuration-prop \
     --set check-schema:false
   ```

   Result:

   The system generates an admin action alert that provides alternate options to disabling schema checking.

```
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]:
```

2. To continue the process of disabling the schema checking instead of following one of the alternate options, press Enter.

Managing matching rule uses

Matching rule use definitions map certain attribute types with a matching rule definition for extensible match filters.

Extensible match filters allow clients to search using distinguished name (DN) components, such as (`ou:dn:=engineering`), or using an OID number, such as (`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 isn't 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 uses can be specified with existing schema components and don't require additional code for implementation.

The following formal specification for attribute types is provided in RFC 4512, section 4.1.4.

```
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 aren't 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.
```

Viewing 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

- To view the PingDirectory server's published list of matching rule uses that use the operational attribute `matchingRuleUse`, run `ldapsearch`.

  > **Note**
  >
  > The multi-valued operational attribute `matchingRuleUse` publishes the definitions on the PingDirectory server, if any. The attribute is stored in the subschema subentry.

  Example:

  ```
  $ bin/ldapsearch --baseDN cn=schema --searchScope base \
    "(objectclass=*)" matchingRuleUse
  ```

Managing DIT content rules

Directory information tree (DIT) content rules provide a way to precisely define what attributes might 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 following:

- Mandatory and optional attributes that entries contain
- The set of auxiliary object classes that entries can be part of
- 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 don't require additional code for implementation.

On the PingDirectory server, only one DIT content rule can be defined for an entry in the structural object class.

The following formal specification for attribute types is provided in RFC 4512, section 4.1.6.

```
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 aren't 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.
```

Viewing DIT content rules

Steps

- To view the `dITContentRules` attribute, run `ldapsearch`.

> **Note**
>
> `dITContentRules` is a multi-valued operational attribute that publishes the definitions on the PingDirectory server, if any. The attribute is stored in the subschema subentry.

Example:

```
$ 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 you are naming 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 can be associated with at most one name form definition.

Name form definitions

Name forms can be specified with existing schema components and don't require additional code for implementation.

The following formal specification for attribute types is provided in RFC 4512, section 4.1.7.2.

```
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 aren't 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.
```

Viewing name forms

Steps

- To view the `nameForms` attribute, run `ldapsearch`.

> **Note**
>
> `nameForms` is a multi-valued operational attribute that publishes the definitions on the PingDirectory server. The attribute is stored in the subschema subentry.

Example:

```
$ bin/ldapsearch --baseDN cn=schema --searchScope base \
"(objectclass=*)" nameForms
```

Managing DIT structure rules

Directory information tree (DIT) structure rules define which entries might be superior or subordinate to other entries in the DIT.

Together with name forms, DIT structure rules determine how relative distinguished names (RDNs) are added together to make up distinguished names (DNs). Because DITs don't 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, 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 don't require additional code for implementation.

The following formal specification for attribute types is provided in RFC 4512, section 4.1.7.1.

```
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 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.
```

Viewing DIT structure rules

Steps

- To view the `dITStructureRules` attribute, run `ldapsearch`.

> **Note**
>
> `dITStructureRules` is a multi-valued operational attribute that publishes the definitions on the PingDirectory server. The attribute is stored in the subschema subentry.

Example:

```
$ bin/ldapsearch --baseDN cn=schema --searchScope base \
  "(objectclass=*)" dITStructureRules
```

About managing JSON attribute values

The PingDirectory server supports a JSON object attribute syntax that 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

You should always specify the `EQUALITY` matching rule as `jsonObjectExactMatch` in the schema definition. Using the `jsonObjectFilterExtensibleMatch` isn't 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 `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 can 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 isn't 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"` *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" })'
```

`"containsField"` *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" })'
```

`"greaterThan"` *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 several constraints for the fields included in JSON objects stored in values of a specified attribute type.

Defining constraints on JSON object fields allows administrators to set requirements on what values a JSON field can take to ensure consistency across entries.

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 to a length specified by the user.
- Restricting values of numeric fields with upper and lower bounds.

> **Note**
>
> Only new entries are subject to the new constraints. Existing data that doesn't match newly-defined JSON constraints can still be decoded and managed by the server. Attempts to alter existing entries with non-compliant JSON objects might require fixing those objects to conform to the new constraints.

The 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. `attribute-type` is 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. The default value is `true`.

If `allow-unnamed-fields` is `true` JSON objects can have fields that are not referenced, and no constraints are imposed on those fields. If `allow-unnamed-fields` is `false`, JSON objects can only have the defined fields.

> **Note**
>
> Unless a schema definition is configured with `allow-unnamed-fields` set to `false`, only information about fields whose values should be indexed or tokenized is required. To ensure that clients cannot store invalid values, you should define any other expected fields. .

As with the 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 might violate those constraints. JSON schema constraints are also enforced for data provided in an LDIF import, and entries containing JSON objects that violate these constraints are 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 includes a period, then that period must be escaped with a backslash.

Property	Description
`value-type`	Specifies the expected data type for the target field. The available values are: ***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 can contain additional elements that define constraints for the fields. ***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 the target field 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. The field constraint can be present with a value of one of the single strings listed below. If the constraint is absent, a value of `prohibited` is assumed. Values are: ***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 can be an array with zero or more values of the specified type, or it can be a single value of the specified type. ***prohibited*** The target field must not be an array and can only be a single value of the specified type.

Property	Description
`allow-null-value`	Specifies whether the target field can have a value of `null` as an alternative to the specified `value-type`. If the field constraint is present, `allows-null-value`'s value must be either `true` or `false`. If the constraint is absent, a default of `false` is assumed. `allow-null-value` has no effect if the `value-type` is `null`.
`allow-empty-object`	Specifies whether empty objects are permitted if the value of the target field is a JSON object or an array of JSON objects. If the field constraint is present, the value must be either `true`, where the object can have zero or more fields, or `false`, where the object can have one or more fields. If the constraint is absent, a default of `false` is assumed.
`index-values`	Specifies whether values of the target field should be indexed in backends. If the field constraint is present, the value must be either `true` (the field should be indexed) or `false` (the field should not be indexed). If the constraint 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 can appear before the entry ID list for that value is no longer maintained. If the field constraint is present, the value must be an integer greater than or equal to 1. If the constraint is absent, the server uses 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 is used. If a cache mode of `cache-keys-only` is configured, priming only loads 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 the field constraint is present, the value is either `true` (field values should be tokenized) or `false` (field values should not be tokenized). If the constraint is absent, a value of `false` is assumed. If `true`, the `value-type` must be `string`.
`allowed-values`	Specifies an explicit set of values allowed in the target field . If the field constraint is 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. `allowed-values` can only be used with a `value-type` of `string`. If the constraint is absent, any value can be used as long as it satisfies all other defined constraints.

Property	Description
`allowed-value-regular-expression`	Specifies a regular expression that the target field must match If the field constraint is present, the value must be a single string or an array of strings representing valid regular expressions (which might require escaping to represent in JSON). Any attempt to use a value that does not match one of the provided regular expressions is rejected. `allowed-value-regular-expression` can only be used with a `value-type` of `string`. If the constraint is absent, values are not required to match any regular expression.
`minimum-numeric-value`	Specifies the lower bound for values of the target field If the field constraint is present, the value must be a single number and can be an integer. Any attempt to use a value that is less than this number is rejected. `minimum-numeric-value` can only be used with a `value-type` of `integer` or `number`. If the constraint is absent, no minimum numeric value applies.
`maximum-numeric-value`	Specifies the upper bound for values of the target field If the field constraint is present, the value must be a single number and can be an integer. Any attempt to use a value that is greater than this number is rejected. `maximum-numeric-value` can only be used with a `value-type` of `integer` or `number`. If the constraint is absent, no maximum numeric value applies.
`minimum-value-length`	Specifies the minimum number of characters allowed for values of the target field If the field constraint is present, the value must be an integer. Any attempt to use a value with fewer characters than this number is rejected. `minimum-value-length` can only be used with a `value-type` of `string`. If the constraint is absent, no minimum length applies.
`maximum-value-length`	Specifies the maximum number of characters allowed for values of the target field If the field constraint is present, the value must be an integer. Any attempt to use a value with more characters than this number is rejected. `maximum-value-length` can only be used with a `value-type` of `string`. If the constraint is absent, no maximum length applies.
`minimum-value-count`	Specifies the minimum number of elements that must be present in an array If the field constraint is present, the value must be an integer. Any attempt to use an array with fewer elements is rejected. `minimum-value-count` can only be used with an `is-array` value of `required` or `optional`. If the constraint is absent, no minimum array value count applies.
`maximum-value-count`	Specifies the maximum number of elements that must be present in an array If the field constraint is present, the value must be an integer. Any attempt to use an array with more elements is rejected. `maximum-value-count` can only be used with an `is-array` value of `required` or `optional`. If the constraint is absent, no maximum array value count applies.

When writing JSON objects in a local database 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 and 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

Adding constraints to JSON attributes

About this task

To create and configure JSON attribute constraints:

Steps

- Run `dsconfig` with the `create-json-attribute-constraints` option.

  Example:

  In this example, a JSON attribute constraint is defined on the `appjson` attribute type. This constrains values of the `appjson` attribute to be JSON objects. Because `allow-unnamed-fields` is set to false, those JSON objects can only have fields for which there is a corresponding JSON field constraints definition.

  > **Note**
  >
  > `appjson` is meant to be replaced by the name of the user's desired attribute type, as defined in the directory schema. For more information, see About managing JSON attribute values.

  ```
  $ bin/dsconfig create-json-attribute-constraints \
    --attribute-type appjson \
    --set enabled:true \
    --set allow-unnamed-fields:false
  ```

  Example:

  In this example, a JSON field constraint object is defined for the `email.verified` field. The `email.verified` field must be present and must take a boolean value.

```
$ bin/dsconfig create-json-field-constraints \
  --attribute-type appjson \
  --json-field email.verified \
  --set value-type:boolean \
  --set is-required:true \
  --set index-values:false \
  --set tokenize-values:false \
  --set allow-null-value:true
```

Example:

In this example, a JSON field constraint object is defined for the `email.type` field. The `email.type` field must be present and must take a value of `home`, `work`, or `other`.

```
$ bin/dsconfig create-json-field-constraints \
  --attribute-type appjson \
  --json-field email.type \
  --set value-type:string \
  --set is-required:true \
  --set index-values:false \
  --set tokenize-values:true \
  --set allowed-value:home \
  --set allowed-value:other \
  --set allowed-value:work \
  --set allow-null-value:false
```

Example:

In this example, a JSON field constraint is defined for the `email.value` field. The `email.value` field must be present and must take a string value that matches the specified regular expression.

```
$ 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:false \
  --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 sign-on attempts, account lockout, and other aspects of password and account maintenance on the PingDirectory server. The 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 don't require a password policy.

Viewing password policies

Password policies enforce 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
- 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 PingDirectory server provides three out-of-the-box password policies that you can apply to your entries or as templates for configuring customized policies:

Default password policy

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.

Root password policy

The root password policy is enforced for the default root user, which uses a stronger password storage scheme (PBKDF2 instead of the salted 256-bit SHA-2 scheme) and requires that a root user provide their current password to select a new password.

Secure password policy

The secure password policy provides a more secure option than the default policy that makes use of several features, including password expiration, account lockout, last sign-on time and last sign-on IP address tracking, password history, and several password validators.

> **Caution**
>
> Using the Secure password policy as-is might notably increase write load in the server by requiring updates to password policy state attributes in user entries and by requiring users to change passwords more frequently. In environments where write throughput is a concern (including environments spread across multiple data centers requiring replication over a WAN), it might be useful to consider whether the policy should be updated to reduce the number of required entry updates.

Viewing password policies

About this task

To view the list of password policies configured on the PingDirectory server:

Steps

- Do one of the following:

  Choose from:

  - Run the `dsconfig` tool in either interactive or non-interactive mode.
  - Use the admin console.

    Example:

    The following example demonstrates the process for obtaining a list of defined password policies in non-interactive mode.

    ```
    $ bin/dsconfig list-password-policies
    ```

    Result:

    ```
    Password Policy         : Type    : password-attribute : default-password-storage-scheme
    ------------------------:---------:--------------------:------------------------------
    Default Password Policy : generic : userPassword       : Salted SHA-256
    Root Password Policy    : generic : userPassword       : PBKDF2
    Secure Password Policy  : generic : userPassword       : PBKDF2
    ```

Viewing a specific password policy

About this task

To view a specific password policy:

Steps

- Do one of the following:

  Choose from:

  - Run the `dsconfig` tool.
  - Use the admin console.

    Example:

    This example uses `dsconfig` to view the default password policy that applies to all uses for which no specific policy is configured.

    ```
    $ bin/dsconfig get-password-policy-prop \
      --policy-name "Default Password Policy"
    ```

    Result:

```
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 PingDirectory server provides several configurable properties that you can use 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 their own password, then the server consults this property for the user's password policy and ensures 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 PingDirectory server after their account is created.

`force-change-on-reset`

Specifies whether users are required to change their passwords after they're 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 can remain idle (the associated user does not authenticate to the server) before that user is locked out. For accounts that don't have a last sign-on time value, the password changed time or the account creation time is used. If that information is not available, then the user isn't allowed to authenticate.

> **Note**
>
> The server should be allowed to run for a period of time with last sign-on time tracking enabled, such as values for both `last-login-time-attribute` and `last-login-time-format` properties to ensure that users have a last sign-on 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 can attempt to bind with the wrong password before that user's account becomes locked either temporarily (in which case the account is automatically unlocked after a configurable length of time) or permanently (in which case an administrator must reset the user's password before the account is 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 they must choose a new one. The value can be expressed in seconds (s), minutes (m), hours (h), days (d), or weeks (w). You can specify a minimum length of time before the user can 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 they receive 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. You can use this setting 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 can mean either using a secure communication channel between the client and the server, or using a Simple Authentication and Security Layer (SASL) mechanism that doesn't 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, you can set a **`failed-bind-response-delay`** configuration property 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 won't 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 sign-on history for both successful and failed sign-on attempts.

You can maintain this history by count or duration, and you can configure the history separately for successful and failed sign-on attempts. Each record in the sign-on history contains the following:

- The time of the sign-on attempt
- The client IP address
- The authentication method
- The reason for failure for failed attempts

You can 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 configure the server to maintain each attempt separately, or to only update the history at most once per day.

Recent sign-on history is disabled by default. You can enable and configure the recent sign-on history in password policy. You can retrieve sign-on history with the `get recent login history` control, available in the LDAP SDK or with the `ldapsearch` and `ldapmodfy` commands. The recent sign-on history is also available in the `ds-pwp-state-json` JSON attribute, the password policy state extended operation, and the `manage-account` command-line tool.

You can enable and configure the recent sign-on history with the following password policy configuration properties:

`maximum-recent-login-history-successful-authentication-count`

The maximum number of records that the server maintains about recent successful authentications.

`maximum-recent-login-history-successful-authentication-duration`

The maximum length of time for which the server maintains information about recent successful authentications.

`maximum-recent-login-history-failed-authentication-count`

The maximum number of records that the server maintains about recent failed authentication attempts.

`maximum-recent-login-history-failed-authentication-duration`

The maximum length of time for which the server maintains information about recent failed authentication attempts.

`recent-login-history-similar-attempt-behavior`

The behavior that the server exhibits 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 only maintains one record for any given date with the same values for all non-timestamp fields, and it uses 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 reflects the most recent attempt on that date.

This is the default behavior.

`maintain-every-attempt`

Indicates that the server maintains a separate record for every attempt, regardless of how similar it is to a previous attempt although duplicate attempts within the same millisecond might not be preserved.

For more information, see Replication considerations.

`update-at-most-once-per-day`

Indicates that the server only maintains 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 sign-on history, but the value of the timestamp field might 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 sign-on 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 you configure the password policy to maintain a successful count of 10 and a successful duration of 30 days, then a user who successfully authenticates on more than 10 dates in a 30-day period would be capped at 10 records. 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 are maintained if you want to maintain records by duration rather than count. Because multiple records can be generated for the same user on the same date, if something is different (such as a different IP address or authentication method), there is technically no limit to the number of records that can be generated when using only a duration-based cap. To mitigate this, you can specify a maximum count to place an upper bound on what information the server maintains for a given user.

The password policy state for a given user is only updated when that user attempts to authenticate to the server. 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.

If you configure the server to maintain recent sign-on history for successful authentication attempts, then it keeps a record of the most recent attempt even if the attempt occurs outside of the maximum duration.

> **Note**
>
> If the `recent-login-history-similar-attempt-behavior` is set to `update-at-most-once-per-day`, it keeps an attempt from the same date as the most recent attempt.

If you configure the server to maintain a history of failed attempts, then it provides information about the most recent failed attempt even if it is older than the maximum duration.

If you configure the password policy to maintain a recent sign-on 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 can 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 you can use to retrieve the recent sign-on history. The value returned in this operation is 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 you can use to clear the `get recent login history` for a user.

Replication considerations

Each sign-on 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 sign-on attempts was 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 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 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) might cause the `additional-attempt-count` value to be incremented to the same value on each of those servers.

> **Note**
>
> If the concurrent attempts occur within the same millisecond, then only one of them is preserved and any others are lost. If the attempts do not occur within the same millisecond, then you can infer the correct value during internal processing, but there might still be corner cases in which the server loses information about one or more attempts.

Get Recent Login History control

You can use a pair of request and response controls to obtain the recent sign-on history.

The request control, which has an OID of 1.3.6.1.4.1.30221.2.5.61 and no value, can be included in a bind request to indicate that the server should return the recent sign-on 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 sign-on history.

The JSON object will have one or both of two top-level fields:

successful-attempts

This field is present if the server is configured to maintain a history of successful attempts, and its value is an array of JSON objects with information about those successful attempts.

In particular, each of those objects contains the following fields, as used in the `ds-pwp-recent-login-history-json` attribute:

- `timestamp`
- `client-ip-address`
- `authentication-method`
- `additional-attempt-count`

failed-attempts

This field is present if you configure the server to maintain a history of failed attempts, and its value is an array of JSON objects with information about these failed attempts.

In particular, each of those objects contains the following fields, as used in the `ds-pwp-recent-login-history-json` attribute:

- `timestamp`
- `client-ip-address`
- `authentication-method`
- `failure-reason`
- `additional-attempt-count`

The response control is only returned if the server is configured to maintain a recent sign-on 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. By ensuring that the request control does not have a value and that the response control value is a simple string, this information is readily accessible to applications using other APIs.

Modifying an existing password policy

Modify an existing password policy to meet your company's requirements.

About this task

> **Tip**
>
> Keep your password policies synchronized across all PingDirectory servers and PingDirectoryProxy server instances.

To modify the configuration for any defined password policy:

Steps

- Choose one of the following methods:

Choose from:

- Run the `dsconfig` tool in interactive or non-interactive mode.
- Use the admin console.

  Example:

  The following example sets some of the properties presented in the previous section for the default password policy using `dsconfig`.

```
$ 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 new password policies that meet your organization's requirements.

You can create any number of password policies in the PingDirectory server using either the `dsconfig` tool (in interactive or non-interactive mode) or the admin console.

> **Tip**
>
> You can find best practices for creating password policies in Password policy tips to improve performance.

Creating a new password policy

Steps

- To create a new password policy:

  Choose from:

  - Run `dsconfig` in interactive or non-interactive mode.
  - Use the admin console.

    Example:

    This example demonstrates 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

Rather than a user automatically inheriting the default password policy, you can assign a user to a particular password policy by including the `ds-pwp-password-policy-dn` operational attribute in that user's entry with a value equal to the distinguished name (DN) of the desired password policy for that user. This operational attribute is explicitly included in a user's entry, or 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 an LDIF file that adds the `ds-pwp-password-policy-dn` attribute with the password policy DN you want to assign to that user.

   Example:

   This example creates the 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. To apply the modification to the user's entry, run `ldapmodify`.

   Example:

   For this example, the file used is `assign.ldif`.

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

Assigning a password policy using a virtual attribute

About this task

You can automatically assign a custom password policy for a set of users using a virtual attribute. You can configure the virtual attribute so that it uses a range of criteria for selecting the entries for which the virtual attribute should appear.

Steps

1. Create an LDIF file, which you can use to add a group to the server.

   Example:

   ```
   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
   ```

2. To add the entries to the server, run the `ldapmodify` tool.

   Example:

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

3. To create a virtual attribute, run `dsconfig`.

   Example:

   This virtual attribute adds the `ds-pwp-password-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. To verify that a user in the group contains the assigned password policy distinguished name (DN), run the `ldapsearch` tool.

   Example:

   ```
   $ bin/ldapsearch --baseDN dc=example,dc=com "(uid=user.0)" \
   ds-pwp-password-policy-dn
   ```

   Result:

```
dn: uid=user.0,ou=People,dc=example,dc=com
ds-pwp-password-policy-dn: cn=Demo Password Policy,cn=Password Policies,cn=config
```

Deleting a password policy

You can delete a password policy with the `dsconfig` tool running in interactive or non-interactive mode.

About this task

> **Note**
>
> You can also use the admin console to delete a password policy.

Steps

- To delete a password policy, run the `dsconfig` command with the `delete-password-policy` subcommand.

  Example:

```
$ 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 PingDirectory server:

- Perform a modify operation which replaces the value of the password attribute (often `userPassword`).

  > **Note**
  >
  > In some configurations, when a user attempts to change their own password, it might 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**
  >
  > If a user is changing their own password, it might be necessary to provide the current password value. If the new password is acceptable to all configured password validators, the server provides a new password, or it can automatically generate a new password for the user.

> **Note**
>
> Regardless of the mechanism used to change the password, all password values should be provided in cleartext rather than pre-encoded, and the user must have sufficient access control rights to update the password attribute in the target user's entry.
> When one user attempts to change the password for another user, the requester must have the `password-reset` privilege.

Validating a password

About this task

A new password is valid if it:

- Meets the server's password requirements
- Is assigned a password policy
- Passes user authentication

The server can display requirements for a password change to users. You can use the `get-password-quality-requirements` extended operation to retrieve information about the requirements, which you can forward to an end user before an attempted password change. You can also use these requirements 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.

You can configure password validators with user-friendly messages that describe the password requirements, and error messages that display if a proposed password does not satisfy those requirements. The server will automatically generate these messages if they are not provided in the configuration.

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's 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 is placed in a "must change password" state where the user is allowed to bind, but any other operations the user attempts 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.

Steps

1. Create a password validator.

> **Note**
>
> Password validator properties include `validator-requirement-description` and `validator-failure` messages.

Example:

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
```

2. To make the newly created password validator take effect, assign it to a password policy.

Example:

```
$ dsconfig set-password-policy-prop \
  --policy-name "Default Password Policy" \
  --set "password-validator:Minimum 5 Characters Password Validator"
```

3. To validate the password, authenticate as a user.

> **Note**
>
> When a user authenticates, password validation processing is performed so that the server has access to the user's clear-text password.

Retiring a password

About this task

An account password can be retired and rotated out of service instead of being invalidated. Retiring a password 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.

Steps

- To enable password retirement, set the `password-retirement-behavior` and `maximum-retired-password-age` properties in the password policy configuration.
- To manually retire an account password or purge a password that has been retired, run the `ldapmodify` and `ldappasswordmodify` tools with subcommands `-- retireCurrentPassword` and `--purgeCurrentPassword`.

> **Note**
>
> To use these commands on an account, enable the **`password-retirement-behavior`** subcommand on the password policy that governs the account.

Changing a user's password using the Modify operation

Steps

- Use `ldapmodify` to change a user's password by replacing the `userPassword` attribute.

  Example:

```
$ 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

About this task

To request that the Password Modify extended operation be used to modify a user's password:

Steps

- Run the `ldappasswordmodify` tool.

  Example:

```
$ bin/ldappasswordmodify --authzID dn:uid=jdoe,ou=People,dc=example,dc=com \
  --newPassword newpw
```

Using an automatically-generated password

Steps

- To automatically generate a new password for a user, run the `ldappasswordmodify` tool.

  Example:

```
$ bin/ldappasswordmodify --authzID "u:user.1"
```

  Result:

```
The LDAP password modify operation was successful
Generated Password: fbi27oqy
```

Enabling YubiKey authentication

Users can authenticate with YubiKey devices, which generate secure one-time passcodes (OTPs) with the `UNBOUNDID-YUBIKEY-OTP SASL` mechanism.

A YubiKey device generates a different password for every authentication attempt, and that OTP is sent to a validation service to ensure that it is genuine and has not been used in an earlier authentication attempt.

> **Note**
>
> It is possible to use the OTP as the only proof of identity, but you should combine it 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

> **Note**
>
> 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 sign-on

You can enable authentication involving credentials that do not reside in, or cannot be forwarded to or validated by, the PingDirectory server (such as social sign-on through Facebook, Google, or Twitter) with the `UNBOUNDID-EXTERNALLY-PROCESSED-AUTHENTICATION SASL` mechanism.

The bind request does not include any credentials, and authentication with this mechanism does not actually change the state of the underlying client connection. The server behaves as if the bind request included the retain identity request control, 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 contro` l in the bind request to obtain information about any password policy state issues that might cause the PingDirectory server authentication attempt to fail. You can include the password policy request control to obtain certain password policy state warnings and errors or to look for the password expired or password expiring controls in the bind response.

Managing user accounts

Manage user accounts with the `manage-account` tool.

About this task

The PingDirectory server provides a user management utility, the `manage-account` tool, that provides a means to quickly view and manipulate several password and account policy properties for a user or group of users.

The server also hosts the Self Service Account Manager project at https://github.com/pingidentity/ssam, which is a customizable web application that allows users to perform their own account registration, profile updates, and password changes.

> **Note**
>
> The project is for testing and development purposes and is not a supported PingDirectory server application.

Steps

- Unlock a user account with the `manage-account` tool.
- To enable a disabled account, contact the administrator for assistance.

> **Note**
>
> Password resets are not used.

Returning the password policy state information

Steps

- To get information about the account's password policy, run the `manage-account` tool.

Example:

```
$ bin/manage-account get-all \
  --targetDN uid=user.1,ou=People,dc=example,dc=com
```

Result:

```
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

- To determine whether a user's account has been disabled, run the `manage-account` tool with the `get-account-is-disabled` subcommand.

  Example:

  ```
  $ bin/manage-account get-account-is-disabled \
    --targetDN uid=user.1,ou=People,dc=example,dc=com
  ```

  Result:

  If the account has been disabled, you receive the following message.

  `Account Is Disabled: true`

Disabling an account

Steps

- To disable a user's account, run the `manage-account` tool.

  Example:

  ```
  $ bin/manage-account set-account-is-disabled \
    --operationValue true --targetDN uid=user.1,ou=People,dc=example,dc=com
  ```

  Result:

  You receive the following message. `Account Is Disabled: true`

Enabling a disabled account

Steps

- To enable a user's account, run the `manage-account` tool with the `clear-account-is-disabled` subcommand.

  Example:

  ```
  $ bin/manage-account clear-account-is-disabled \
    --targetDN uid=user.1,ou=People,dc=example,dc=com
  ```

  Result:

  You receive the following message. `Account Is Disabled: false`

Assigning the manage-account access privileges to non-root users

Assign access rights to the non-root admin user.

About this task

Non-root users, such as `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 might 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.

Steps

1. Create a non-root user admin account, such as `uid=admin,dc=example,dc=com`.
    1. Grant the `password-reset` privilege to the account.

    Learn more in steps 1 and 6 of Setting up a single administrator account.
2. To view the account status for an account, run the `manage-account` tool.

    Example:

    ```
    $ bin/manage-account get-all \
      --targetDN uid=user.0,ou=People,dc=example,dc=com
    ```

    Result:

    The system displays the following information for the account.

    ```
    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.

    Example:

The following allows access to manage accounts to a helpdesk user. Depending on the configuration requirements, this user might 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. To disable an account, run the `manage-account` tool.

   Example:

   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
   ```

   Result:

   You receive the following message. `Account Is Disabled: true`

5. To view the presence of the `ds-pwp-account-disabled` operational attribute in the entry, run the `ldapsearch` tool.

   Example:

   ```
   $ bin/ldapsearch --baseDN dc=example,dc=com "(uid=user.0)" "+"
   ```

   Result:

   The system displays the following information.

   ```
   dn: uid=user.0,ou=People,dc=example,dc=com
   ds-pwp-account-disabled: true
   ```

Bypassing password policy evaluation

You can bypass password policy evaluations when performing operations on accounts other than your own.

About this task

The PingDirectory server supports the use of a `bypass-pw-policy` privilege, which can skip password policy evaluation for operations on a per-user basis. If a user has this privilege, then they are allowed to perform operations on user entries that would normally be rejected by the password policy associated with the target entry.

> **Note**
>
> The `bypass-pw-policy` privilege does not have any effect for bind operations.

Any user with this privilege will be permitted to perform operations against other users that would otherwise be rejected under the constraints associated with that user's password policy, such as:

- Setting a pre-encoded password
- Setting a new password that wouldn't be accepted by one or more password validators
- Setting a new password that already exists in a user's password history

> **Note**
>
> These restrictions can also be circumvented on a per-operation basis using the password update behavior control. If you have a set of users that should be subject to lesser or differing constraints than another set of users, you can create a new password policy with the desired constraints, if any, and assign it to the appropriate users. Learn more about assigning password policies to users.

Steps

- To add the `bypass-pw-policy` privilege to a user entry, run the `ldapmodify` tool with the `bypass-pw-policy` subcommand.

  Example:

```
$ bin/ldapmodify
dn: uid=user.1,ou=People,dc=example,dc=com
changetype: modify
add: ds-privilege-name
ds-privilege-name: bypass-pw-policy
```

Managing password validators

A password validator is a password policy component you can use 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 they all must consider a proposed new password acceptable before it is allowed.

Password validators

The PingDirectory server offers several types of password validators.

You can use the Server SDK to create custom password validators with the constraints 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. You can configure the validator to look in all attributes or in a subset of attributes. It can perform forward and reverse mapping and reject values that 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`. While it can be used to create a custom validator, it 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, 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 might be stronger than a shorter password containing a mix of uppercase and lowercase letters, numbers, and symbols. This is based on the Gibson Research Corporation Password Haystacks concept.
Length-Based Password Validator	Ensures that the number of characters in the proposed new password is within an acceptable range. Both a maximum and minimum number of characters can be specified.
Pwned Passwords Password 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.
Regular Expression Validator	Ensures that a proposed password either matches or does not match a given regular expression.

Password Validators	Description
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"). You can configure the validator to operate in a case-sensitive or case-insensitive manner, and you can also define custom sets of equivalent characters, such as defining 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** For this password validator to be effective, you must have access to the user's current password. If this password validator is 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 admin console to configure or modify any password validators.

After you define 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

- To view the set of password validators defined in the server, run the `dsconfig` tool.

Configuring the Attribute Value Password Validator

Steps

1. To edit the existing default configuration for the Attribute Value Password Validator, run the `dsconfig` tool.

   Example:

   In this example, the configuration 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.

   Example:

```
$ 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`).

   Example:

```
$ bin/ldappasswordmodify --authzID "uid=user.0,ou=People,dc=example,dc=com" \
  --newPassword user.0
```

   Result:

```
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. To edit the existing default configuration, run the `dsconfig` tool.

   Example:

   This example changes the requirement for special characters by making them optional in a password and adds a requirement to include at least two digits in the password. 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.

Example:

```
$ 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).

   Example:

   This 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. To edit the existing default configuration, run the `dsconfig` tool.

   Example:

   In this example, the required minimum number of characters in a password is set 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.

   Example:

   ```
   $ 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.

   Example:

   ```
   $ bin/ldappasswordmodify \
     --authzID "uid=user.0,ou=People,dc=example,dc=com" --newPassword abcd
   ```

   Result:

```
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 Pwned Passwords Password Validator

The server is preconfigured with an instance of the Pwned Passwords Password Validator.

About this task

Use the `dsconfig` tool to configure the Pwned Passwords Password Validator.

Steps

1. Run the `dsconfig` tool to update an existing password policy to use the Pwned Passwords Password Validator, replacing *[PASSWORD_POLICY_NAME]* with the name of your password policy.

   ```
   dsconfig set-password-policy-prop \
            --policy-name "[PASSWORD_POLICY_NAME]" \
            --add "password-validator:Pwned Passwords"
   ```

2. Test the validator by submitting a password that is known to be compromised; for example, `password`.

Configuring the Regular Expression Password Validator

About this task

Use the `dsconfig` tool to configure the Regular Expression Password Validator.

Steps

1. Use `dsconfig` to create a regular expression password validator.

   Example:

   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 is 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 password validator.

   Example:

```
$ bin/dsconfig set-password-policy-prop \
  --policy-name "Default Password Policy" \
  --set "password-validator:Regular Expression"
```

3. Test the regular expression password validator by submitting a password that meets the requirements.

> **Note**
>
> To meet the password requirements, make sure your password contains one number, one lowercase letter, and one uppercase letter.

Example:

```
$ bin/ldappasswordmodify \
  --authzID "uid=user.0,ou=People,dc=example,dc=com" --newPassword baaA1
```

Result:

```
The LDAP password modify operation was successful
```

4. Test a password that does not meet the password requirements.

Example:

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
```

Result:

```
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

You can configure the Repeated Character Password Validator with the `dsconfig` command. This validator ensures that user passwords don't contain character fragments, such as strings of repeated characters like "aaaaaa" or "aaabbb."

Steps

1. To edit the existing default configuration, run the `dsconfig` tool.

Choose from:

- Set the maximum consecutive length of any character.

> **Note**
>
> For the following example, the maximum consecutive length of any character is set to 3. The validator rejects any passwords with 4 or more consecutive characters, such as "baaaa1" or "4eeeeb", etc.

```
$ bin/dsconfig set-password-validator-prop \
  --validator-name "Repeated Characters" \
  --set max-consecutive-length:3
```

- **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).**

> **Note**
>
> You can 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, such as "aaa", "bbb", "ccc", "abc", or "123". However, a password, such as "12a3", is acceptable.

```
$ 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.**

   Example:

```
$ bin/dsconfig --no-prompt set-password-policy-prop \
  --policy-name "Default Password Policy" \
  --set "password-validator:Repeated Characters"
```

3. **To test the Repeated Character Validator, submit a password that has more than the maximum allowable length of consecutive characters.**

   Example:

   For this example, the faulty password submitted is `baaa1`.

```
$ bin/ldappasswordmodify \
  --authzID "uid=user.0,ou=People,dc=example,dc=com" \
  --newPassword baaa1
```

   Result:

```
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

Use the `dsconfig` tool to configure the Similarity-Based Password Validator.

Steps

1. To edit the existing default configuration, run the **`dsconfig`** tool.

   Example:

   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.

   > **Note**
   >
   > 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.

   Example:

   ```
   $ 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, such as 2.

   > **Note**
   >
   > The **`ldappasswordmodify`** command requires the `--currentPassword` option when testing the Similarity-Based Password Validator.

   Example:

   ```
   $ bin/ldappasswordmodify \
     --authzID "uid=user.0,ou=People,dc=example,dc=com" \
     --currentPassword abcde --newPassword abcdd
   ```

   Result:

   ```
   The LDAP password modify operation failed with result code 49.
   ```

Configuring the Unique Characters Password Validator

Use the `dsconfig` tool to configure, update, and test the Unique Characters Password Validator.

Steps

1. To edit the existing default configuration, run the `dsconfig` tool.

   Example:

   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 "Unique Characters" --set min-unique-characters:3
   ```

2. Update an existing password policy to use the Unique Characters Password Validator.

   Example:

   ```
   $ 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, such as 3.

   Example:

   ```
   $ bin/ldappasswordmodify \
     --authzID "uid=user.0,ou=People,dc=example,dc=com" \
     --newPassword aaaaa
   ```

   Result:

   ```
   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.

Every server in the topology can handle write requests, with the replication component performing immediate synchronization with other members. The replicated server environment ensures that Lightweight Directory Access Protocol (LDAP) clients can seamlessly fail over to another server instance.

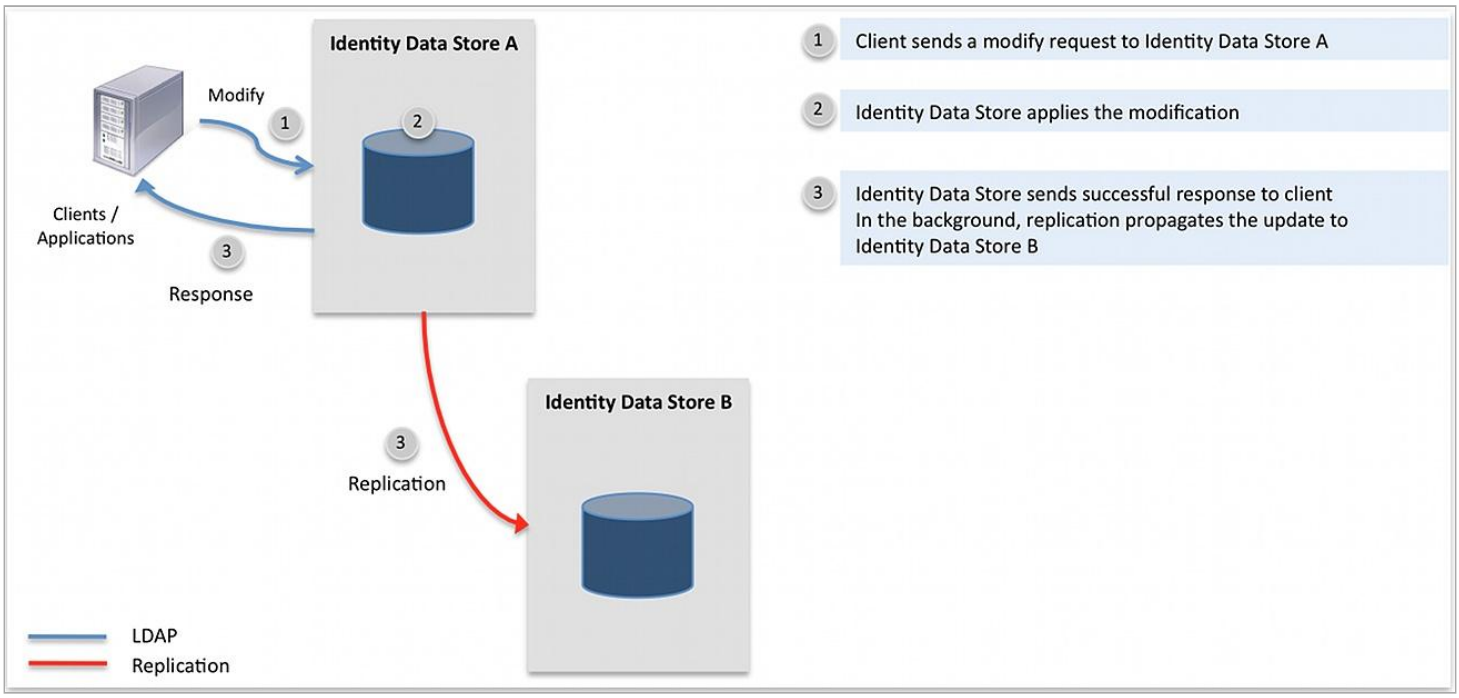
Overview of replication

Replication is a data synchronization mechanism that ensures that updates made to a database are automatically duplicated 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 PingDirectory server instances participating in replication are writable so that LDAP clients can perform updates at any of these server instances. These updates are automatically propagated to the other servers in the background 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.



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 because of a planned maintenance or unplanned outage. Servers that are temporarily unavailable receive updates when they become available again.

Improved Search Performance

Search requests can be directed to any PingDirectory server participating in replication, which improves search performance over systems that only access single servers.

> **Note**
>
> Replication does not improve write throughput because 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 and synchronization are distinct functions from one another and are not interchangeable.

Replication is not a general purpose synchronization function because it creates replicas with exact copies of the replicated data. By contrast, synchronization can:

- Transform data between two different directory information tree (DIT) structures
- Map attribute types
- Synchronize subsets of branches and specific object classes

> **Note**
>
> For data migration, use synchronization instead of replication unless you are adding cloud directories to an existing PingDirectory cluster.

The differences between replication and synchronization are as follows:

Replication doesn't support differing DIT structures

The distinguished name (DN) of replicated entries must be the same on all servers. In some situations, you might want 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 could 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 doesn't support differing FIPS-compliance modes

Replication can't occur between server instances running in FIPS-compliant mode and server instances running in non-FIPS-compliant mode. Learn more about the Differences between FIPS-compliant and non-FIPS-compliant modes.

Replication can't map attribute types or transform attribute values

Some situations might require you to map attribute types or transform attribute values when synchronizing data from one server to another. Replication doesn't support attribute type mappings or attribute value transformations.

Replication doesn't support fractional replication

Replication can't be configured to replicate a subset of the attribute types from the replicated data set. Synchronization fully supports this feature.

Replication doesn't support sparse replication

Replication can't be configured to replicate only entries with a particular object class. 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 received updates by replication, which is considered an internal operation. While trust between servers is established between two endpoint servers, synchronization doesn't require full control of the data. Disparate server system endpoints can be synchronized, such as a PingDirectory server and a relational database management system (RDBMS) database endpoint with each fully in control of its own data.

If replication doesn't meet your data synchronization requirements, consider using PingDataSync server instead, which provides the versatility and robust performance required for most production environments.

Replication terminology

The following terms have specific definitions as they apply to replication.

Replication Terminology

Term	Description
Assured Replication	Data is guaranteed to replicate before the response is returned to the client. This is useful for applications that require immediate access to replicated data or require data consistency over response time.
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 in milliseconds since 00:00:00 UTC, January 1, 1970 • The replica ID • The sequence number
Conflict	Client updates made at different replicas affecting the same entry might 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 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 eventually be 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 enablement 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`.

Term	Description
Historical Data	Historical Data are records of attribute value changes as a result of an LDAP Modify operation. Historical data are 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 can have similar performance characteristics when accessed from this PingDirectory server or reside in the same data center or city. A separate location setting can be defined for each data center, such as Austin, London, Chicago, and so forth. Location settings are used in the selection of the WAN Gateway server.
Modify Conflict	A conflict between two LDAP Modify operations. Modify conflicts are automatically resolved.
Naming Conflict	Any conflict other than Modify Conflicts. Naming conflicts typically include an operation that changes the DN of the target entry, creates an entry, or deletes an entry at one replica.
Replica	The component in the PingDirectory server that handles interaction with a single replication domain. For example, the `dc=example,dc=com` replication domain within the `userRoot` backend is a replica.
Replica ID	The unique identifier of a replica. Replica ID is automatically set 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. Each replica in the state can have a maximum of one change number. The Replication Server component uses the replica state to determine which updates the Replica has not yet received.
Replication	An automated background process that pushes PingDirectory server data changes to all other replicas.

Term	Description
Replication Changelog	A backend maintained independently by each replication server 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 is the external changelog that clients can access. It is located at `<server-root>/db/changelog`. • The Replication Changelog Backend is the changelog that replication servers use. It is located at `<server-root>/changelogDB`, and is not accessible by clients or 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 PingDirectory server process that is responsible for propagating server data changes to and from replicas. Each PingDirectory server instance participating in replication is also running a replication server.
Replication Server ID	The unique identifier of a replication server. The replication server ID is 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. Each replica in the topology can have a maximum of one change number. 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. The sequence number increases incrementally for every update at the replica. The initial value of the sequence number is 1. The number is stored as a 32-bit integer, but only positive values are used. The sequence number can roll over.
WAN Gateway	The designated replication server that assumes the WAN Gateway role within a collection of 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.

Term	Description
WAN Gateway Priority	The configuration setting that determines which replication server assumes the WAN Gateway role. The replication server with the highest WAN Gateway Priority in a location assumes the role of the WAN Gateway. Servers can have a WAN Gateway Priority value from 1 to 10, with 1 being the highest priority. A server with a WAN Gateway priority value of 0 will never be a gateway.

Replication architecture

This section introduces the major elements of replication in PingDirectory server.

Eventual consistency

Replication is based on an eventual-consistency model where the updates propagated through a connected network of servers will eventually be consistent on all servers over a short period of time.

In a typical update operation, a client application updates an entry or group of entries on a PingDirectory server with an `ADD`, `DELETE`, `MODIFY`, or `MODIFY DN` operation. After processing the operation, the PingDirectory server returns an LDAP response while simultaneously propagating the update to the other servers in the replicated topology. This simultaneous processing model allows the client to continue submitting update requests without waiting for a replication completion response from the server.

To support this processing model, replication never locks the targeted entries at the other PingDirectory server instances before an update can be made locally. This means that the replicated servers might have an inconsistent view of the targeted entry for a 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 because clients can propagate a change without having to wait for replication. The rate of update operations remains the same no matter how many PingDirectory servers participate in replication.

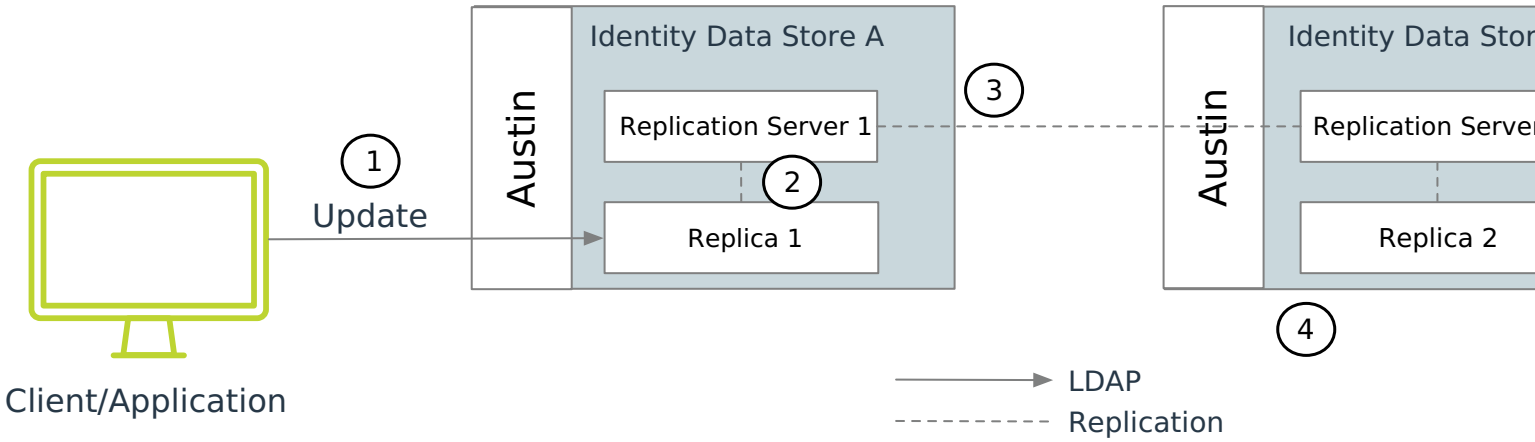
Alternatively, you can configure assured replication for specific write requests to require local or global consistency, across data center locations, before a response is returned to the client. For more information, see Configuring assured replication.

Replicas and replication servers

Each PingDirectory server has an embedded replication server that is responsible for transmitting updates to other replication servers. Each replicating base distinguished name (DN), such as `cn=schema, dc=example,dc=com`, has a separate component called a replica. Each replica connects to the embedded replication server running within the PingDirectory server Java virtual machine (JVM) process.

When a client application updates an entry on the PingDirectory 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 PingDirectory server. The replication server on other PingDirectory servers then passes the change to the appropriate replica, which in turn applies the change to its backend after performing conflict resolution.

The following figure shows the standard setup.



Replicas and Replication Servers

1. The client or application sends a modification request to Identity Store A. The identity store applies the change.
2. The Replica backend in Identity Store A sends an update message to the Replication Server in Identity Store A.
3. The Replication Server in Identity Store A pushes the update to the Replication Server in Identity Store B.
4. The Replication Server in Identity Store B applies the update the to the Replica backend in Identity Store B.

Authentication and authorization

The PingDirectory server uses a simple authentication model to authorize replication servers.

After it is authenticated, the remote PingDirectory server is fully authorized to exchange replication messages with the local PingDirectory server. There is no other access control in place.

Authentication in the replication protocol is based on public key cryptography using TLS client certificate authentication. The certificate used for authentication is stored in the `ads-truststore` backend of the PingDirectory server.

During replication setup, the command-line utility distributes public keys to all PingDirectory servers to establish trust between the servers and to enable TLS client authentication.

Logging

The access log messages in the PingDirectory server indicate if the update was received by replication and includes the corresponding change number. This allows the administrator to track which server the update originated from.

Replication deployment planning

Consider the following factors before deploying replication in a production environment:

Minimizing Replication Conflicts

If two different clients attempt to create an entry with the same distinguished name (DN) at the same time against different servers, it's possible that both client requests will succeed. This triggers the server to send a conflict alert. You can minimize these conflicts by understanding the client traffic pattern beforehand. PingDirectory's Assured Replication feature and the PingDirectoryProxy server can both assist in minimizing conflicts.

Learn more at Configuring assured replication and PingDirectoryProxy.

Replication Purge Delay

Adjust the default one-day replication purge delay 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 with longer replication purge delays.

Define a minimum value for the replication purge delay. Make sure the value is consistent across all servers.

Learn more at Modifying the replication purge delay.

Domains

All servers in a replication topology must have the same domains.

> **Note**
>
> Because the PingDirectory replication topology assumes that all domains are on all servers, it is not suitable for environments with an inconsistent presence of domains across the server topology.

Location

If your deployment incorporates multiple data regions, you should configure the PingDirectory servers with location information.

To set locations, use the `dsconfig create-location` and `dsconfig set-global-configuration-prop` commands.

The PingDirectory server cannot determine LAN boundaries automatically, so incorrect location settings can result in undesired WAN communication. By default, replication compresses all traffic between PingDirectory servers in different locations.

Set up the locations before enabling replication. The `dsreplication enable` command prompts for location information if you have forgotten to define the property.

User-defined LDAP

PingDirectory servers participating in replication must 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 PingDirectory servers.

Disk space

Replication increases the disk space required for the PingDirectory server.

The Replication Changelog backend keeps changes from all PingDirectory 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 dependent on the rate and size of changes in the replication topology and the Replication Changelog purge delay.

Memory

Replicated PingDirectory server instances require slightly more memory than a standalone server.

All of the items discussed in Disk space have an impact on the amount of memory the PingDirectory 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, you should keep system clocks in sync within the deployment.

Communication ports

The replication server component in each PingDirectory server listens on a TCP/IP port for replication communication.

This replication server port, typically 8989, must be accessible from all PingDirectory servers participating in replication. The server-to-server communication channel is kept alive using a heartbeat, which occurs every 10 seconds. This traffic prevents firewalls from closing connections prematurely.

The replication command-line utility (`dsreplication`) requires access to all PingDirectory servers participating in replication. This includes the LDAP or LDAPS port of the servers.

> **Tip**
>
> When configuring firewalls, keep these communication requirements in mind.

Hardware load balancers

Replication allows write operations to be directed to any PingDirectory server in the topology. Distributing write operations in a round-robin fashion could introduce conflicts. In particular, distributing a series of LDAP `add`, `delete`, and `modify DN` operations targeting the same distinguished name (DN) in quick succession can cause conflicts that require manual intervention.

The Assured Replication feature can help prevent conflicts created by the same client application. For more information, see Configuring assured replication.

Consider using server affinity with the load balancer that either associates a client IP address or a client connection with a single PingDirectory server instance at a time.

For load balancing LDAP traffic, consider using the PingDirectoryProxy server. The Server Affinity feature in PingDirectoryProxy server enables replication-friendly load balancing. For more information, see Configuring server affinity.

PingDirectoryProxy

To facilitate replication-friendly load balancing, consider using PingDirectoryProxy in every replication deployment.

The PingDirectoryProxy server can automatically adapt to conditions in the backend PingDirectory server using health checks and route traffic accordingly.

For example, the PingDirectoryProxy server can re-route traffic from PingDirectory servers with a large backlog of replication updates.

Displaying the server information for a replication deployment

Steps

- To display the server information for a replication deployment, run the `dsreplication status` command with the `--displayServerTable` option.

```
$ bin/dsreplication status --displayServerTable
```

Result:

```
          --- 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

- To display the status information for a replication deploymnent, 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 PingDirectory servers ensures that changes within the replicated base distinguished name (DN) are automatically propagated to all other PingDirectory servers in the topology.

Configuration changes are not replicated by default but can be through the use of server groups. Each PingDirectory server should be configured according to the deployment design.

Overview

To interface with the replication topology, the PingDirectory server provides a command-line utility, `dsreplication`, that should 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 might 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 step to set up a PingDirectory 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. For more information, see Command-line interface.

- Initialize data from source server to all servers in the topology.

  Run the `dsreplication initialize` subcommand for every target server that needs a copy of the data from the source server.

- Verify the replication topology.

  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, use the `dsreplication` tool in interactive mode.

The `dsreplication` tool uses the following format, including some important subcommands listed in About 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, and the public key used during the replication authentication process.

  If 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 see their replication server configuration updated with the information of the new replication server.
- A replication domain is created for the requested base distinguished names (DNs). If 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 ensures that a uniform schema is present in the replication topology.
- Initialization must be performed for the enabled base DNs.

> **Important**
>
> The `dsreplication enable` subcommand removes all existing topology administrative users except for the default `uid=admin` that is provided to enable replication. Administrative users must be re-added after enabling replication.

Initialization

Replica initialization transfers a copy of the backend containing the replication domain to a target server. You should perform this 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`, `port 1`. 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. Existing data on the target server replica is lost and the backend containing the base DN is taken offline on the target server during the initialization.

Binary copy

The database copy method involves copying database backup files from the source PingDirectory server to one or more target servers. The PingDirectory 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 must then be transferred to the target servers 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` followed by `bin/encryption-settings import` must be performed on the `encryption-settings` backend.
- When using database copy to initialize a server that has been offline longer than the replication purge delay, the database copy of the `replicationChanges`, `schema`, and `adminRoot` backends are required.

Import/export LDIF

The import/export LDIF method is the most flexible approach, as it works well in environments where target replicas are configured differently than the backend containing the data (for example, the target replica has different indexes than the source server). The process involves exporting the backend data as LDIF from the source server and importing the LDIF into the target servers. Replicas can be initialized locally while the server is offline or online through remote access. For more information, see Importing and exporting data.

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. Two replicas containing the same data have the same generation ID.

When replication is operating correctly, all of the replicas for each replicated base DN have the same generation ID. The generation ID is stored on each replica as the operational attribute `ds-sync-generation-id`, as in the following 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, 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 ensures that the ID does not need to be computed the next time the replica is loaded.

Deploying a basic replication topology

Set up a two-server replication topology.

About this task

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 PingDirectory server with 2000 sample entries.

   Example:

   ```
   $ ./setup --acceptLicense --baseDN "dc=example,dc=com" --ldapPort 1389 \
     --rootUserPassword pass --sampleData 2000 --no-prompt
   ```

2. Install the second PingDirectory server either on a separate host or the same host as the first, but with a different LDAP port.

   Example:

   ```
   $ ./setup --acceptLicense --baseDN "dc=example,dc=com" --ldapPort 1389 \
     --rootUserPassword pass --no-prompt
   ```

3. To configure a replication topology, from the first server, run the `bin/dsreplication` command in interactive mode.

   Example:

```
$ bin/dsreplication
```

4. From the `Replication Main Menu`, select `Manage the topology`.

   Example:

```
>>>> 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, select `Enable Replication`.

   Example:

```
>>>> 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. From the `Enable Replication` menu, read the setup introduction. To continue the enable process, enter `c`.

7. Enter the LDAP connection parameters for the first of the two server replicas that you are configuring.

   1. Enter the host name or IP address of the server.

   2. Enter the type of LDAP connection to the server.

   Choose from:

   - LDAP
   - LDAP with SSL

- **LDAP with StartTLS**

1. **Enter the LDAP listener port for the server.**

> **Note**
>
> If you are a root user, you see port 389 as the default. Others see port 1389.

2. **Authenticate as a root distinguished name (DN), such as `cn=Directory Manager`.**

> **Note**
>
> Later in the process, you are prompted to set up a global administrator and password. The global administrator is the user ID that manages the replication topology group.

8. **Repeat steps 7a-7d for the second server replica.**

9. **When the `dsreplication` tool checks for the base DN on both servers, press enter to select the default base DN.**

> **Note**
>
> To enable replication, data must be present on at least one of the servers.

Example:

```
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, in most cases, a base DN was not set up on one of the PingDirectory servers, or the backend is disabled.
>
> ```
> There are no base DNs available to enable replication between the two servers.
> ```

10. **When you are prompted to set up entry balancing using the PingDirectoryProxy server, press enter to accept the default value.**

> **Note**
>
> Learn more in the PingDirectoryProxy Server Administration Guide.

Example:

```
Do you plan to configure entry balancing using the Directory Proxy Server? (yes / no) [no]:
```

11. **Enter the replication port for the first replica.**

> **Note**
>
> The default replication port is 8989.
> The port you enter must be free.

12. If you are prompted to enter a location, press enter to accept the default and set up a location for the first server. Enter the name of the server's location.

> **Note**
>
> You are prompted for the location if you did not pre-define a location setting for the server.

Example:

```
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
```

13. Repeat steps 9-12 for the second PingDirectory server.

> **Note**
>
> If you did not pre-define a location setting for the second server, you are prompted to enter location information for the server.

14. Set up the global administrator user ID and a password for this account.

> **Note**
>
> The default global administrator user ID is `admin`.
> The global administrator user ID manages the PingDirectory servers used in the replication topology.

Example:

```
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:
```

15. Return to the `Replication Main Menu` and enter the number corresponding to initializing data over the network.
16. To initialize data on a single server, from the `Initialize Replica Data over the Network` menu, select `Initialize`, and enter `c` to continue.
17. To specify a server in the replication topology, enter the following information for the first server:
    - Host name or IP address

- LDAP connection type
- LDAP port
- Global admin user ID and password

18. 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.

19. Select the base DN to initialize.

> **Note**
>
> In most cases, the base DN for the root suffix is replicated. In this example, `dc=example,dc=com`.

20. Select the second server to have its data initialized, and enter the global admin user ID and password for the target server.

> **Note**
>
> Any data present on the target server is over-written.

21. To confirm that you want to initialize data on the target server, press enter.

    Example:

```
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]:
```

    Result:

    The server displays a `Base DN initialized successfully` message.

22. On the `Initialize Replica Data Over Network` menu, enter `b` to back out one level to the main menu.

23. To view the replication status, on the `Replication Main Menu`, enter the host name or IP address.

    Example:

```
     --- 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

The following example creates a 4-server topology spanning two data centers.

In the example, the four servers are already installed and have locations Austin and Budapest 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`, `port 1`.
> 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. To generate the sample data on the first server, use the **`make-ldif`** tool.

   Example:

   ```
   $ bin/make-ldif --templateFile config/MakeLDIF/example-10K.template \
     --ldifFile ldif/10K.ldif
   ```

2. To import the data, select a server from which to import data and to be the source for future initialization to other servers, and then follow the import steps.

   1. Stop the server.
   2. To import the sample LDIF to the server, run **`import-ldif`**.
   3. Start the server.

      Example:

      ```
      $ bin/stop-server
      $ bin/import-ldif --backendID userRoot --ldifFile ldif/10K.ldif
      $ bin/start-server
      ```

3. To enable replication, select a specific server and run **`dsreplication enable`** three times.

   > **Note**
   >
   > For the first invocation, create the replication topology administrator with the name **`admin`**.

   Example:

```
$ 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. To initialize the other servers, the `dc=example,dc=com` replicas, from the server you imported the data into with **`import-ldif`**, run **`dsreplication initialize`**.

   Example:

   For this example, initialize **`budapest02`** from **`budapest01`** to minimize the WAN transfers.

```
$ 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. To view the replication state, run **`dsreplication status`**.

   Example:

```
$ 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 PingDirectory 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.

> **Note**
>
> 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 distinguished name (DN).

Steps

1. **To set the LDAP Connection Handler to explicitly listen on the server's host name address, run `dsconfig` with the `set-connection-handler-prop` option.**

   Example:

   ```
   $ 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. **To 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, run `dsconfig` with the `set-identity-mapper-prop` option.**

   > **Note**
   >
   > You must do this to map the admin account to the Kerberos realm.

   Example:

   ```
   $ 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. **To invoke replication enable, authenticate as the existing Kerberos `authid`, and then run `dsreplication enable`.**

   > **Note**
   >
   > No bind DNs and passwords are required to authenticate because you are using SASL binding. However, the new replication admin user requires a password at creation time, and you should use a strong random password. After SASL is working, you no longer have to provide this random password.
   > You must use 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.

   Example:

```
$ 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. To initialize data on the remote server, run `dsreplication initialize`.

   Example:

```
$ 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. To delete the temporary `userPassword` from the `replication.admin` entry, create an LDIF file.

   Example:

   In this example, the file is named `remove-password.ldif` and contains the following.

```
dn: cn=replication.admin,cn=Administrators,cn=topology,cn=config
changetype: modify
delete: userPassword
```

6. To apply the modifications, use the `ldapmodify` tool.

   Example:

```
$ ./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. To check the topology's status, run `dsreplication status`.

> **Note**
>
> After the admin account and mappers are created, you can use the `--saslOption useTicketCache=true` and `--saslOption requireCache=true` properties instead of a password for all **`dsreplication`** commands.

Example:

```
$ 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.

There is a small window of time where updates are not all present on the replicas while the changes are replicating through the system. However, there are 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 uses 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. The level of assurance applied to each operation can be explicitly requested by the client, specified by the server configuration using the Replication Assurance Policy, or both.

Assured replication is supported by client requests directly to PingDirectory server, through a PingDirectoryProxy server, or both.

About the Replication Assurance Policy

Assured replication uses a Replication Assurance Policy to define the properties needed to ensure that replication has 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 that, 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, you should associate request or connection criteria with a policy to apply replication assurance appropriately.

PingDirectory server contains a default Replication Assurance Policy that is enabled with the local-level processed-all-servers assurance level. 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 (such as 1) has precedence over policies with larger `evaluation-order-index` values (such as 2, 3, 4, and so forth). 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 among multiple Replication Assurance Policies that match against an operation. The smaller value has precedence.

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` returns 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` returns 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 have the assurance level of `processed-all-servers` and `add` operations received on server B have the assurance level of `received-any-server`.

About assured replication

When implementing assured replication, make note of the following.

Item	Description
Client Controls	The client can include an assured replication request control with each operation. This control allows the client to specify bounds on assurance levels, override the timeout assigned by the matching replication assurance policy for the associated operation, or both. The server always honors these request controls. Learn more in About the Assured Replication Controls.
Directory Proxy Server	Replication assurance policies are not supported on PingDirectoryProxy. Replication client controls are passed through to the underlying PingDirectory server backend.
Schema Backend Replication	The schema backend is not supported by assured replication. Replication assurance policies that include criteria to match this backend are 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.

Item	Description
Global Configuration Properties	PingDirectory server provides two configurable global configuration properties that determine the timing of the assurance source and maximum number of replication backup updates recognized as an available source. **`replication-assurance-source-timeout-suspend-duration`** Specifies the amount of time a replication assurance source is suspended from assurance requirements if it experiences an assurance timeout. While suspended, the source is excluded from assurance requirements for all operations originating on this PingDirectory server. This avoids the situation of repeated timeouts caused by degraded or offline servers. The default time 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 PingDirectory server with more than the configured threshold backlog updates, it is excluded from assurance requirements for all operations originating from the server until it completes at least one assurance successfully, indicated by the server receiving an update acknowledgment message from it within the timeout window. The default maximum number of updates is 1000.

Configuring assured replication

About this task

It is common for all servers to have the same policy. The following example, which demonstrates the configuration of various assured replication policies, assumes that three servers are configured on localhost, on ports 1389, 2389, and 3389.

> **Note**
>
> In this example, each server has a default Replication Assurance Policy with no assurance levels set.

Steps

1. To create request criteria for `add` operations on server 1, run `dsconfig` with the `create-request-criteria` option.

> **Note**
>
> This request criteria is used to match any `add` operation with the Replication Assurance Policy that is configured in the following step.

Example:

```
$ bin/dsconfig create-request-criteria \
  --criteria-name Adds \
  --type simple \
  --set operation-type:add
```

2. **To make all add operations assured with a level of `processed-all-servers` on server 1, set up the Replication Assurance Policy using `dsconfig` and specify the `Adds` request criteria configured in the previous step.**

> **Note**
>
> The `processed-all-servers` level indicates that all local servers in the topology must complete replay of the replication update before the response is sent to the client.

Example:

```
$ 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 steps 1-2 for `modify` operations.**

> **Note**
>
> 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.

Example:

```
$ 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. **To set up the `Adds` and `Mods` request criteria and a Replication Assurance Policy on server 2, repeat steps 1-3.**

Example:

```
$ 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
```

Note

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.

5. **To list the policies on server 1 to confirm that they exist, run `dsconfig` with the `list-replication-assurance-policies` option.**

   Example:

   ```
   $ bin/dsconfig list-replication-assurance-policies
   ```

6. **Repeat step 5 for server 2 and server 3.**

   Server 3 should only show the Default Replication Assurance Policy.

7. **To check the Replication Assurance counters on all servers before any `add` or `modify` operation, use `ldapsearch`.**

   Note

   The counters should be set to zero. These counters are on the replica server where the policy is matched and assigned.

   Example:

   For this example, 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
```

8. To check the Replication Summary table on all of the servers, use `ldapsearch`.

   Example:

   For this 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
```

9. To add an entry to server 1 on the PingDirectory server, use `ldapmodify`.

   The counters should match the newly added entry to the `Adds Processed All Locally` policy and complete assured.

   Example:

```
$ 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
```

10. Perform a `modify` of an entry under `dc=example,dc=com` on server 1.

    The counters should match the `modify` operation to the `Mods Processed All Locally` policy and the operations should complete assured.

    Example:

```
$ 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
```

Result:

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`, `modifyDN`, and certain extended requests to indicate the optimal level of replication assurance for the associated operation. The OID for this control is 1.3.6.1.4.1.30221.2.5.28 and can have a criticality of either `TRUE` or `FALSE`.

For more information, see the LDAP SDK Javadoc for the `AssuredReplicationRequestControl` class.

Managing the topology

The following sections describe common topology management operations.

When enabling or disabling replication within a topology that contains multiple product versions, you must run the **`dsreplication`** tool from the server root location of a member of the topology that has the oldest product version.

Adding a server to the topology

Before you begin

You must:

- Have an existing PingDirectory server topology.
- Ensure that more than 50% of servers in the topology and the new server are online.

About this task

The commands are identical for initial enablement 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, the server being initialized must have a copy of the `encryption-settings` backend from the source server.

Steps

1. To enable replication for the base distinguished name (DN), or base DNs, run `dsreplication enable` using an existing server as `host1` and the new server as `host2`.

   Example:

   ```
   $ 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
   ```

2. (Optional) To compare the configurations between the two hosts used in the `dsreplication enable` command, run `config-diff`.

   > **Note**
   >
   > Ensure settings are consistent across the topology and are also consistent with the new system.

   Example:

   ```
   $ bin/config-diff --sourceLocal \
     --targetHost austin03.example.com \
     --targetBindDN "cn=directory manager" \
     --targetBindPassword pass --targetPort 1389
   ```

Replacing the data for a replicating domain

About this task

In case the data for the entire replication domain, such as the backend, needs to be replaced, perform the following steps:

Steps

1. With all servers online, run `dsreplication pre-external-initialization` against any server in the topology.

Example:

```
$ bin/dsreplication pre-external-initialization --hostname austin01.example.com \
   --port 1389 --baseDN dc=example,dc=com --adminUID admin \
   --adminPassword password --no-prompt
```

Result:

Replication is stopped for the domain. No writes are made by clients to any of the servers.

2. To replace the data for `dc=example,dc=com`, use `import-ldif`.
    1. (Optional) To overwrite the existing data for the domain, use the `--overwriteExistingEntries` option.
    2. (Optional) To ensure that the input LDIF is free of any replication attributes, use the `--excludeReplication` option.

       Example:

       In this example, you perform the `import-ldif` with the server offline using the `--excludeReplication` and `--overwriteExistingEntries` options.

       ```
       $ bin/import-ldif --ldifFile new-data.ldif --backendID userRoot --excludeReplication --
       overwriteExistingEntries
       ```

3. To initialize the other servers in the topology, run `dsreplication initialize` using the server that has the new data as the source host.

   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. From any server in the topology, run `dsreplication post-external-initialization` once.

   > **Note**
   >
   > All servers in the topology must be online.

   Example:

   ```
   $ bin/dsreplication post-external-initialization --hostname austin01.example.com \
      --port 1389 --baseDN dc=example,dc=com --adminUID admin \
      --adminPassword password --no-prompt
   ```

Disabling replication and removing a server from the topology

Before you begin

More than 50% of the servers not being removed from the topology must be online during the process.

About this task

When removing a server from the topology, the remaining servers need to be made aware of the change. 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.

Removing a live server

Steps

- To remove a server that is online, run the `dsreplication disable` command from any server.

```
$ bin/dsreplication disable --hostname austin03.example.com --port 1389 \
--baseDN dc=example,dc=com --adminUID admin --adminPassword password \
--no-prompt
```

> **Note**
>
> If the server to remove is online, only one invocation of **`dsreplication disable`** is necessary.

Removing an offline server from the topology

Steps

- To remove a defunct or misbehaving server from the topology while that server is offline, run `remove-defunct-server` from a live server in the topology. Doing this will remove configuration references to the offline server from the topology's perspective. This is useful if the server is broken and cannot start. For example:

```
$ bin/remove-defunct-server --serverInstanceName austin01 \
--bindDN "cn=Directory Manager" --bindPassword password
```

After you have removed the server, you can delete it. However, if you want to keep the server as a standalone server, proceed to Cleaning up an offline defunct server.

> **Note**
>
> The `--ignore-online` option removes an online server cleanly from the topology.

> **Tip**
>
> To speed up the process of removing multiple servers, change the default 10-minute timeout for each server you are taking out of rotation by setting the Java virtual machine (JVM) property `com.unboundid.connectionutils.LdapResponseTimeoutMillis` before you run **`remove-defunct-server`**.

Cleaning up an offline defunct server

About this task

If you have removed a defunct server and want to keep it as a standalone server, then you also need to run the `remove-defunct-server --performLocalCleanup` command on the defunct server while it is offline. This will remove all topology configuration references on the offline server, and make it a standalone server. This is useful if a server is temporarily misbehaving, but you want to add it back to the topology at a later time. The `remove-defunct-server` command will only accept the `--performLocalCleanup` argument if the server is offline.

Steps

- Run the `remove-defunct-server` command on each server that is removed from the topology. For example:

```
$ bin/remove-defunct-server --performLocalCleanup --no-prompt
```

> **Warning**
>
> To remove the defunct server, you must include the `--no-prompt` parameter. If you exclude this parameter, the server won't be removed from the topology. Additionally, the server might attempt to contact additional hosts in the topology, which could result in an error. These are known issues that only apply to `--performLocalCleanup` and will be corrected in a future release.

Advanced configuration

The following sections cover advanced configuration procedures for your company's deployment needs.

Changing the replicationChanges DB location

About this task

If on-disk space issues arise, you can change the `replicationChanges` database (DB) location. The replication changelog database can live outside `<server-root>` and be placed in another location on the file system. To do this, specify the absolute path of the replication changelog directory.

Steps

1. To change the database location for the replication changelog, use the `dsconfig` tool.

> **Note**
>
> By default, the database location is at `<server-root>/changelogDb` .

Example:

The following command sets the replication Changelog Backend to `<server-root>/data/directory/changelogDb` .

> **Note**
>
> Remember to include the LDAP connection parameters, such as host name, port, bindDN, and 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.

   Example:

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

3. Move the DB files.

   Example:

   ```
   $ mv changelogDb /data/directory
   ```

4. Restart the server.

   Example:

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

Modifying the replication purge delay

The replication purge delay specifies the period after which the PingDirectory server purges replication server database changes. Any change that is older than the purge delay is removed from the replication server database regardless of whether the change has been applied.

About this task

Currently, PingDirectory server sets the default purge delay to one day. Administrators can change the default purge delay using the `dsconfig` tool.

> **Note**
>
> To ensure proper replication processing, you must have the same purge delay value set for all replication servers in the topology.

Steps

- To change the purge delay, choose from one of the following options:

  Choose from:

  - Use the `dsconfig` tool.

- Use the `dsconfig` tool in interactive mode:
    1. Open the Advanced objects menu.
    2. Select Replication Server.
    3. Select the replication synchronization provider.
    4. Select the option to change the replication purge delay.

       The property accepts time units of seconds (s), minutes (m), hours (h), days (d), or weeks (w).

Example:

The following command changes the purge delay from the default of one day to two days.

```
$ 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

Configure a single listener-address for the replication server to change the default setting of listening on all addresses.

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, change the address to the host name provided when replication is enabled and set the `listen-on-all-addresses` property to `FALSE`.

The replication server's configuration entry 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 generates 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 PingDirectory server instance and enable replication on port 8989.
2. To see the ports bound for listening on port 8989, run `netstat`.

   > **Note**
   >
   > A port of `*.8989` means that it is listening on all addresses.

   Example:

   ```
   $ netstat -an | grep LISTEN | grep 8989
   ```

3. To disable listening on all addresses for the replication server, run the `dsconfig` tool.

Example:

```
$ bin/dsconfig set-replication-server-prop \
  --provider-name "Multimaster Synchronization" \
  --set listen-on-all-addresses:false
```

4. To see the ports bound for listening on port 8989, run `netstat` again.

> **Note**
>
> A port of `<address>.8989`, such as 10.8.1.211.8989, means that it is listening on the one address.

Monitoring replication

This section covers how to use the PingDirectory server to monitor replication.

The PingDirectory server monitors replication in the following ways:

- Use the `dsreplication` tool with the `status` subcommand to display basic information about the replicated base distinguish names (DNs), the number of entries replicated, as well as the approximate size of replication backlogs at each PingDirectory server.
- Obtain more detailed information about the state of replication through the information exposed in its monitoring framework under `cn=monitor`.
- Use Periodic Stats Logger plugin, which allows collecting replication statistics for profiling server performance. For more information, see Profiling Server Performance Using the Periodic Stats Logger.

Administrators can monitor their replication topologies using several tools and protocols:

- SNMP
- LDAP
- JMX
- The admin console. For more information, see Managing Logging and Managing Monitoring.

Monitoring replication using cn=monitor

The `cn=monitor` branch has several entries that store the replication state of a topology.

`cn=monitor` *branch entries and their descriptions*

cn=monitorentries	Description
Direct LDAP Server `<baseDN><host name:port><serverID>`	**Defines an LDAP server that directly connects to the replication server you are querying. The information in this entry applies to the replication server local to the** `cn=monitor` **entry.** **For more information, see** Summary of the Direct LDAP Monitor information**.**
Indirect LDAP Server `<baseDN><serverID>`	**Defines an LDAP server that connects to another replication server in the topology. While this server connects to the same topology, it does not connect to the replication server you are querying.** **For more information, see** Summary of the Indirect LDAP Server Monitor information**.**
Remote Repl Server `<baseDN><host name:port><serverID>`	**Defines a remote replication server that connects to the local replication server. Information in this entry is in respect to the Replication Server local to the** `cn=monitor` **branch.** **For more 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 PingDirectory server and its communication with a replication server. The replica information sends and receives changes from the replication servers.** **For more information, see** Summary of the Replica Monitor information**.**
Replication Server	**Shows information specific to the Replication Server running.** **For more 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 PingDirectory server, it records those changes in the changelog table.** **For more information, see** Summary of the Replication Server Database Monitor information**.**

cn=monitorentries	Description
Replication Server Database Environment	Shows information for the database environment for the replication server backend as well as the total number of records added to and deleted from the database. For more 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 distinguish name (DN). For more information, see Summary of the Replication Summary Monitor information.
replicationChanges Backend	Shows the backend information for all replication changes. For more information, see Summary of the replicationChanges 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 more information, see Summary of the Replication Protocol Buffer Monitor information.

Configuring missing replication changes

You can configure missing replication changes to prevent server lockdowns that are caused by missing changes. The configuration uses the `missing-changes-policy` enumeration configuration property.

Configuring missing replication changes makes it easier to:

- Recover from difficult cases of missing changes lockdowns.
- Configure replication to favor up-time at the expense of data integrity. Missing changes will optionally be ignored and replication will continue.

> **Note**
>
> There can only be one instance of `missing-changes-policy` per replication server and one instance per replication domain on each replication server.

Missing change types and actions

Missing change types

The missing change types are described in the following table:

Missing change type	Description
`DIRECT`	For server-to-server connections, a contributing replica is found to have missing changes and the missing changes type is not `INDIRECT`. This can include replicas that are deleted but not obsolete, as there isn't sufficient information to determine which server hosted the replica. Also, if the replica has been deleted, it's unlikely that a server will have more recent changes to fix the missing changes lockdown.
`INDIRECT`	For server-to-server connections, a contributing replica is found to have missing changes and the replica is: • In the list of replica IDs for the topology • Hosted by a server not involved in the connection **Note** This excludes deleted replicas.
`OBSOLETE`	The contributing replica with missing changes is obsolete based on the existing definition of obsolete (deleted and all changes older than the purge delay). This change type is only applicable when the global configuration property `replication-purge-obsolete-replicas` is set to true, which it is the default setting. This change type takes precedence over the other change types.
`PREVIOUS`	For local replicas connecting to the replication server, a replica's backend previously had missing changes. This is indicated by a null change number.
`REPLICA`	For local replicas connecting to the replication server, a replica is missing changes relative to its local replication server. This can happen when a backend is restored from an older backup.

Missing change actions

The missing change actions are described in the following table:

Missing change action	Description
`CONTINUE`	Raise an alert but ignore the missing change and continue. The missing changes will be lost. For the PREVIOUS missing changes type, the persistent missing changes state (null change number) will be permanently cleared.
`PAUSE`	Raise an alert and pause update sending for the particular replica that is missing changes from the source server connected to, but otherwise continue replicating data. The pause will remain in effect until the connection for the replica is restarted.
`SERVER`	Use the value from the corresponding missing changes type configuration property on the replication server.

Supported configurations and examples

The allowed `missing-changes-policy` values are described in the following table. Examples of these values used with the `dsconfig` command are also provided.

Value	Description
`MAXIMUM_INTEGRITY`	Enter the missing changes lockdown state in response to missing changes as much as possible. This strongly favors data integrity over availability.
`FAVOR_INTEGRITY`	Enter the missing changes lockdown state in response to missing changes where it makes sense for most customers. This balances data integrity and availability and is the default policy.
`FAVOR_AVAILABILITY`	Provide increased availability when compared to `FAVOR_INTEGRITY` with regard to `PREVIOUS` missing changes.
`MAXIMUM_AVAILABILITY`	Continue in response to missing changes lockdown as much as possible. This strongly favors availability over data integrity.
`SERVER`	Indicates that the replication server's configuration should be used. This makes it possible to have one configuration per replication server that can optionally be overridden for each domain. **Note** `SERVER` is for replication domains only. This is the default value of `missing-changes-policy` for replication domains.

Examples

For maximum integrity, which was the behavior prior to configurable missing changes, the `MAXIMUM_INTEGRITY` missing changes policy can be selected for the replication server:

```
dsconfig set-replication-server-prop --provider-name "Multimaster Synchronization"
--set missing-changes-policy:maximum-integrity
```

For behavior that is similar to `MAXIMUM_INTEGRITY` except that missing changes lockdown is avoided for indirect missing changes, the `FAVOR_INTEGRITY` missing changes policy can be selected for the replication server. This is the default missing changes policy:

```
dsconfig set-replication-server-prop --provider-name "Multimaster Synchronization"
--set missing-changes-policy:favor-integrity
```

To keep the default `FAVOR_INTEGRITY` missing changes policy for all domains except `dc=example,dc=com`, where `MAXIMUM_AVAILABILITY` is to be used:

```
dsconfig set-replication-domain-prop --provider-name "Multimaster Synchronization"
--domain-name dc_example_dc_com --set missing-changes-policy:maximum-availability
```

Replication best practices

This section covers the best practices for replication based on our observations in production environments.

Purging obsolete replicas

About this task

By default, the PingDirectory server retains replicas for every server that has been part of a given topology. Over time, this can lead to an accumulation of obsolete replicas in your topology, which can increase the complexity of replication and negatively impact reliability.

To purge these obsolete replicas by default, you can set the `replication-purge-obsolete-replicas` global configuration property to `true`.

> **Important**
>
> If you set `replication-purge-obsolete-replicas` to `true`, you must make this change on all servers, including new servers as they are added. Failure to do so could cause replicated servers to communicate an inconsistent state across the topology. An inconsistent state can lead to a `replication-missing-changes` alert and a server lockdown because of the missing replication changes.

Steps

- To set `replication-purge-obsolete-replicas` to `true`, run `dsconfig set-global-configuration-prop`, as in the following example:

  Example:

  ```
  bin/dsconfig set-global-configuration-prop \
      --set replication-purge-obsolete-replicas:true
  ```

About the `dsreplication` command-line utility

The following points involve some security practices that apply to replication. For specific questions, 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. Use the recorded commands to develop scripts to set up and configure replication. You can find details about `dsreplication` logs in Replication subcommand logs and exit codes.

Scripts invoking the `dsreplication` utility in non-interactive mode check the return code from the `dsreplication` process. A non-zero return code indicates a failure.

To exclude output messages from the `dsreplication` utility, use the `--quiet` option to suppress them.

By default, the utility fails if one or more warnings are issued during the command execution. To suppress warnings, use the `--ignoreWarnings` option.

> **Note**
>
> This option is required when using **`dsreplication`** with non-fully-qualified host names, such as localhost. Otherwise **`dsreplication`** fails. However, you should avoiding using this flag in production environments.

Concurrent Use

With the exception of using the **`dsreplication`** utility with the `status` subcommand, the **`dsreplication`** subcommands cannot execute 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. For best practices, avoid making concurrent configuration changes.

Status

Using the **`dsreplication`** utility with the `status` subcommand requires the Replication Servers to provide monitoring information. This can lead to a delay before the output of **`dsreplication`** `status` displays. By default, **`dsreplication`** displays the status for all replicated domains with the exception of the special domains of the schema and the server registry.

Select a particular base distinguished name (DN) for **`dsreplication`** `status` if multiple base DNs are configured for replication.

Avoid invoking **`dsreplication`** `status` too often, such as 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 does not refresh every time the monitor is queried.

Do not use the `status` subcommand for collecting performance metrics. It is best to use the Periodic Stats Logger plugin for replication-related information.

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. Use the **`--topologyFilePath`** option to specify a file with a series of hosts and ports in the topology.

For example, 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 generates with the **`manage-topology export`** command.

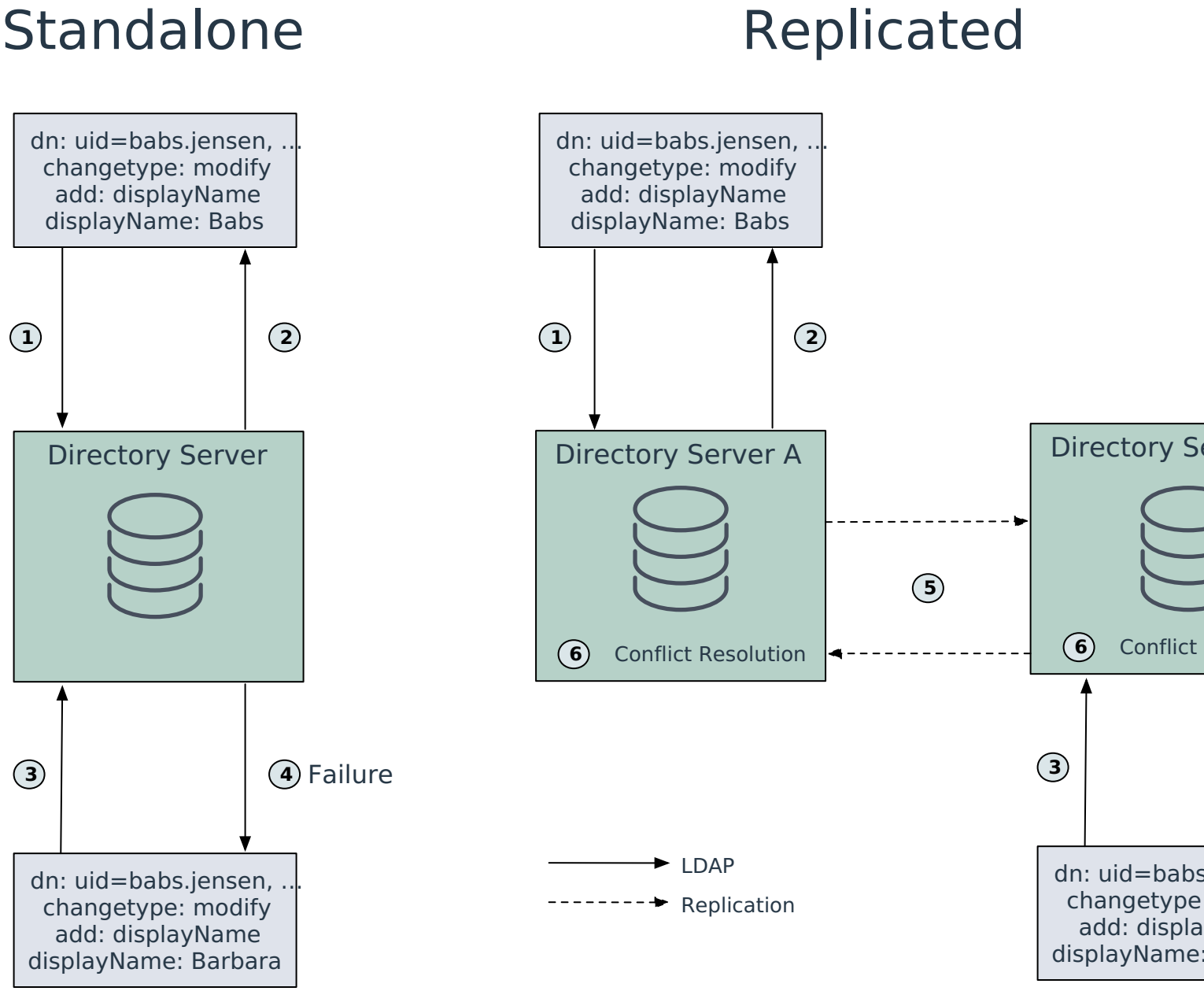
Replication conflicts

This section provides in-depth information about the mechanisms and possible scenarios behind replication conflicts.

Updates to the PingDirectory server entries in a replication topology can happen independently because replication guarantees only eventual consistency and not strong consistency. The eventual consistency model enables applying conflicting changes at different PingDirectory server instances. In most cases, the PingDirectory server can resolve these conflicts automatically and in a consistent manner: for example, all server instances in a replication topology resolving each and every conflict the same way. However, in some scenarios, manual administrative action is required. For any of these unresolved conflicts, the administrator receives notification through 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 apply to a single PingDirectory server instance. For example, if the same entry is added to two different PingDirectory server instances at about the same time, after these operations are replicated, both servers keep only the first entry added.

The following image highlights the differences between standalone versus replicated environments.



Processing steps

1. Client sends LDAP modify request to directory server and succeeds.
2. Client sends another LDAP modify request that conflicts with the first.
3. In the standalone case, the request is rejected. In the replicated case, the request succeeds.
4. Replication propagates the updates (replicated only).
5. Conflict resolution (replicated only): A ignores the update from B, and B sets the attribute value to the value on A.

Types of replication conflicts

There are two types of replication conflicts:

Naming conflicts

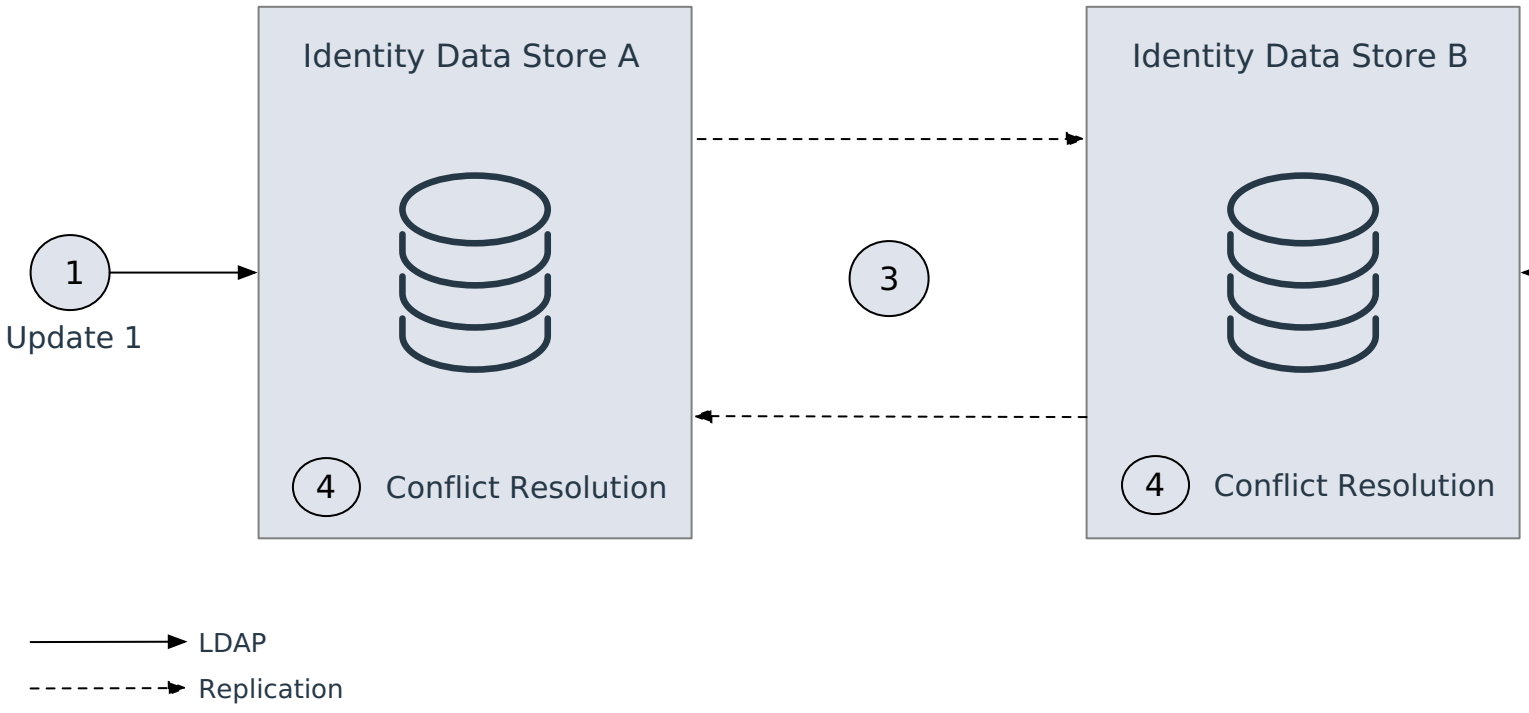
Naming conflicts include operations that cause conflicts with the naming of the distinguish name (DN) of the existing or new entries.

Modification conflicts

Modification conflicts include operations that result in conflicts in the modification of attributes.

Naming conflict scenarios

The following illustration depicts the naming conflict scenario.



Processing steps

1. The Client sends an update (Add, Delete, or Modify DN) request to server A.

2. The Client sends an update (Add, Delete, or Modify DN) request conflicting with the first request to server A.

3. Replication propagates the updates.

4. Conflict resolution takes place.

The following table shows the result of a modification conflict depending on the type of updates that occur. The code does not compare change sequence numbers (CSNs) but applies operations in the order they were received. This might lead to inconsistent replays.

For all of the naming conflict scenarios, assume the following:

- Update 1 was applied at PingDirectory server 1.
- Update 2 was applied at PingDirectory server 2.
- Update 1 occurred shortly before Update 2 so that PingDirectory server 2 received Update 1 after Update 2 was applied.

Naming Conflict Scenarios

Update 1	Update 2	Automatic Resolution?	Result of Conflict Resolution at PingDirectory 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 is 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.

Update 1	Update 2	Automatic Resolution?	Result of Conflict Resolution at PingDirectory server 2 When Update 1 is received
`Add` A	`Delete` of the parent of A	No	The added entry is marked as a conflict entry.
`Add` A	`Add` A with same set of attributes	Yes	The entryUUID of the incoming Add operation applied to the existing entry.
`Add` A	`Add` A with different set of attributes (or values)	No	The existing entry is marked as a conflicting entry and the incoming `Add` is executed.

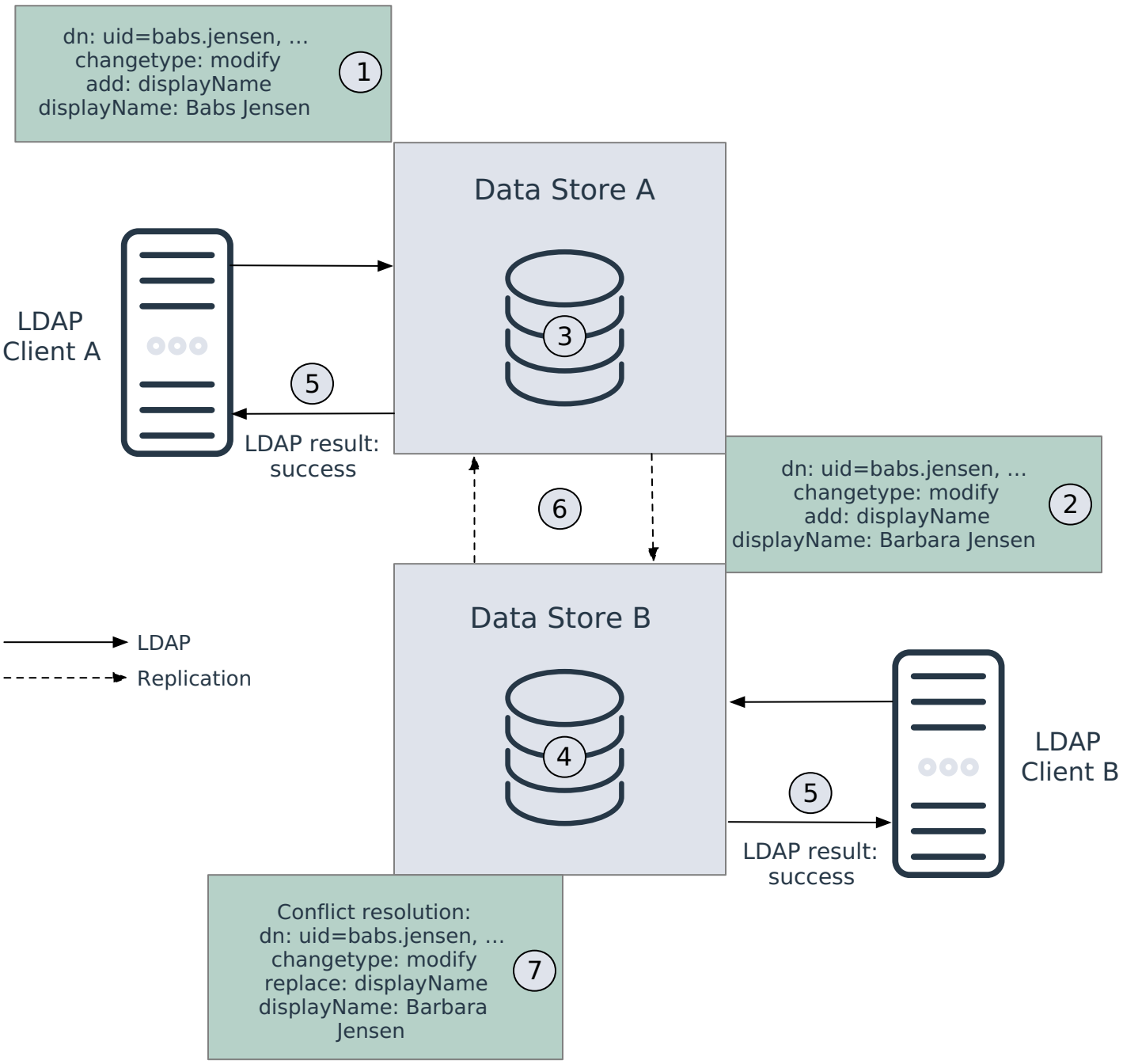
Modification conflict scenarios

Modification conflicts resolve automatically and no manual action is required. The `LDAPModify` 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

> **Note**
>
> Replication does not currently support the increment LDAP modification type.

The following illustration presents a modification conflict scenario.



Processing steps

1. LDAP client A sends a `modify` request to directory server A.
2. LDAP client B sends a conflicting `modify` request to directory server B.
3. Directory server A applies the modification.
4. Directory server B applies the modification.
5. The LDAP clients receive successful responses.
6. Replication propagates the updates.

7. Conflict resolution on directory server A updates the entry to reflect the value on directory server B.

   Conflict resolution on directory server B ignores the update received from directory server A.

The following table shows the result of a modification conflict depending on the type of updates that occur. For all of the operations in the table, assume the following:

- LDAP Modify 1 was applied at PingDirectory server 1.
- LDAP Modify 2 was applied at PingDirectory server 2.
- LDAP Modify 1 occurred shortly before LDAP Modify 2, so that PingDirectory server 2 received LDAP Modify 1 after LDAP Modify 2 was applied.

Modification Conflict Scenarios

Modify 1	Modify 2	Result of Conflict Resolution at PingDirectory 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.

Modify 1	Modify 2	Result of Conflict Resolution at PingDirectory server 2 When Modify 1 is received
`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

This section covers information on troubleshooting your replication deployment.

When troubleshooting, check the log file associated with the subcommand that is producing the error. Learn more about Replication subcommand logs and exit codes.

> **Note**
>
> For replication issues related to certificate trust, see Repairing broken listener certificate trust in replication.

Discovering obsolete replicas

About this task

To avoid entering lockdown mode when upgrading servers in a replicated topology, before upgrading, check the `replicationChanges` database for any obsolete replicas. To perform this check, run the `check-replication-domains` tool, which scans `changelogDb` for all known replication domains and identifies any obsolete replicas still listed as part of a topology.

> **Tip**
>
> As of PingDirectory version 10.2, the risk of lockdown due to obsolete replicas is minimal. If you have already upgraded all servers in a replicated topology to version 10.2 or later, these steps are optional.

Steps

1. Run `check-replication-domains --serverRoot <serverRootDirectory>`.

   > **Note**
   >
   > You can use the `--serverRoot` argument to specify the root directory where the server containing the replication data is installed. If you don't supply this argument, `check-replication-domains` uses the default value of the server where you run the tool.

2. Review the output for any replica IDs listed as `Obsolete`.

   Example:

   The following is an example output from the `check-replication-domains` tool:

```
Server topo-1
[pinguser@topo-1 ~]$ PingDirectory/bin/check-replication-domains --serverRoot PingDirectory/

SERVER           DOMAIN DN                ID
---------------- ------------------------ ------
topo-1           cn=schema                20693 (local)
topo-1           dc=example,dc=com        23135 (local)
topo-2           cn=schema                8371
topo-2           dc=example,dc=com        19233

Server topo-2
[pinguser@topo-2 ~]$ PingDirectory/bin/check-replication-domains --serverRoot PingDirectory/

SERVER           DOMAIN DN                ID
---------------- ------------------------ ------
<unknown>        dc=example,dc=com        7403  OBSOLETE
<unknown>        dc=example,dc=com        7406  DELETED
topo-1           cn=schema                20693
topo-1           dc=example,dc=com        23135
topo-2           cn=schema                8371 (local)
topo-2           dc=example,dc=com        19233 (local)
```

Note

Any replica marked `DELETED` has been deleted from the topology but is not yet obsolete.

Next steps

If you identified any obsolete replicas, purge the obsolete replicas.

Recovering a replica with missed changes

If a server has been offline for a period of time longer than the replication purge delay, you must run the `dsreplication initialize` command to bring the replica into sync with the topology.

Any missed changes are detected at the time of server startup. A missed change is a change that the replica detects it needs, but the change is not found within any other replication server's replicationChanges backend stored in the `/changelogDb` server root path.

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 the server requires a manual backup and restore, perform the following steps, which are equivalent to `dsreplication initialize`.

Performing a manual initialization

About this task

The PingDirectory server provides the tools necessary for backing up and restoring backends, which can be used to manually initialize a replica.

As detailed in the following procedure, you use `<server-root>/bin/backup` to create a backup of the backend containing the replicated base DN. If encryption is enabled for the backend containing the replicated base DN, then you must also make a backup of the `encryption-settings` backend.

When initializing a server that has been offline longer than the `replication-purge-delay`, you must also make backups of the `replicationChanges` and `schema` backends.

You then need to transfer all backup files to the target server(s) and restore them individually using `<server-root>/bin/restore`.

> **Note**
>
> To preserve existing encryption settings, `<server-root>/bin/restore` appends to the `encryption-settings` database as opposed to replacing it.

To manually initialize a server when an online initialization isn't possible:

Steps

1. From another server in the replication topology, back up the `userRoot`, `schema`, `changelog`, and `replicationChanges` backends to the `<server-root>/bak` directory.

   > **Tip**
   >
   > If data encryption is enabled, export the `encryption-settings` backend, because you might need to import one or more encryption settings IDs into the new replica.

   Example:

   ```
   $  <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 changelog --backupDirectory \
      bak/changelog
   $  <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.

   Example:

   ```
   $ scp -r  <source-server-root>/bak/* \
       <user>@<destination-server>:<destination-server-root>/bak
   ```

3. Stop the server.

4. Restore the `userRoot`, `schema`, `changelog`, and `replicationChanges` backends.

> **Important**
>
> If the `encryption-settings` backend was exported, import it before restoring any of the backends.

Example:

```
$  <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/changelog
$  <destination-server-root>/bin/restore --backupDirectory \
   bak/replicationChanges
```

5. Start the server using `bin/start-server`.

Fixing replication conflicts

Replication conflicts occur when an incompatible change to an entry is made on two replicas at the same time. The change processes on one replica and then replicates to the other replica, which causes the conflict. While most conflicts resolve 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 is small, and the command includes the Replication Repair Control specified by OID value `1.3.6.1.4.1.30221.1.5.2`, run `ldapmodify` against the server with the conflict. The Replication Repair Control prevents the change from replicating and enables changing operational attribute values, which are not normally writable.

The following tasks use the Replication Repair Control to fix replication conflicts and apply change only to the server with the conflict. There are two examples provided to fix replication conflicts: one for fixing a modify conflict using the `ldap-diff` tool and the other for fixing a naming conflict.

Fixing a modify conflict

Steps

1. To isolate conflicting entries between two replicas, use the `bin/ldap-diff` tool.

> **Note**
>
> Replace the `sourceHost` value with the server that needs the adjustment.

Example:

The following example uses the tool to search across the entire base distinguish name (DN) for any difference in user attributes and reports the difference in `difference.ldif`.

```
$ 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. To apply changes to the server that contains conflicts, use the `difference.ldif` file in a format compatible with `ldapmodify`.

> **Important**
>
> Run `ldap-diff` command with the `sourceHost` value as the server with conflicts.

Example:

The following is an example of the contents of `difference.ldif` file.

```
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. To correct the entries on the sole server with conflicts, run `bin/ldapmodify`.

Example:

```
$ bin/ldapmodify --bindPassword <password> -J "1.3.6.1.4.1.30221.1.5.2" \
  --filename difference.ldif
```

Fixing a naming conflict

About this task

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`. Because of this conflict, the server returns a replication conflict message. See the following example message.

```
[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'"
```

The PingDirectory 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.

To resolve the conflict:

Steps

1. Search for any entry that has the `ds-sync-conflict-entry` objectclass and returns only the DNs that match the filter.

   Example:

   ```
   $ bin/ldapsearch --baseDN dc=example,dc=com --searchScope sub \
     "(objectclass=ds-sync-conflict-entry)" "1.1"
   ```

   Result:

   The search results display the conflicting entry for uid=user.200.

   ```
   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.

2. Compare the conflict entry with the target entry.
3. Apply the difference in two ways:

   Choose from:

   - Use the `ldapmodify` tool with the Replication Repair Control.

     > **Note**
     >
     > You can also delete the conflict entry using this command.

   - Run `bin/ldapmodify` with the Replication Repair Control to make the fix.

> **Note**
>
> When making changes using the Replication Repair Control, the updates are not propagated through replication. Examine each replica individually, and apply the necessary modifications using the request control.

Example:

```
$ bin/ldapmodify -J "1.3.6.1.4.1.30221.1.5.2" \
  --filename difference.ldif
```

Fixing mismatched generation IDs

About this task

If you receive a warning message that multiple generation IDs were detected for a specific suffix, you must re-initialize one or more replicas. If the warning is presented from a server after an initialization, it could be that the `post-external-initialization` command was not run as part of a global change in data.

Try the following fixes as needed.

Steps

- To re-initialize replicas as part of a global change in data, run the `post-external-initialization` command.
- To fix mismatchd generation IDs, run the `dsreplication` command.
- To warn when any generation IDs are different across the topology, run the `dsreplication` tool with the `status` command.

Replication subcommand logs and exit codes

To preserve important troubleshooting and support information, PingDirectory Server logs `dsreplication` subcommand executions, grouped by name, in the `logs/tools` directory. By default, the server retains the output from the last 10 executions for each subcommand (such as `status`, `enable`, and `initialize`). You can use these logs to review or troubleshoot the execution of a specific subcommand within the retention window.

> **Note**
>
> The retention policy for these logs is not configurable.

If a `dsreplication` subcommand returns a non-standard result code, such as in the event of an error, the server generates a special log file that gets:

- Published to the `logs/tools` directory
- Named according to the result code and timestamp of the execution
- Compressed and retained indefinitely

Check these logs first to more quickly and effectively determine a root cause.

> **Important**
>
> These non-standard logs should be created by the server infrequently and contain valuable information for support technicians. Don't delete these files unless necessary. When you run the `collect-support-data` tool, it generates a support package that includes all non-standard logs and the six most recent logs for each subcommand.

`dsreplication` exit codes

The following table details the exit codes that the server logs for `dsreplication` subcommand executions. Use these codes to troubleshoot errors with replication operations.

Exit code	Details
0	• Successful. • Successful, but no operation was performed.
1	Unable to initialize the arguments.
2	Can't parse the arguments because the user-provided arguments aren't valid or there was an error checking the user data.
3	The user canceled the operation in interactive mode.
4	There was an unexpected error (potential bug).
5	The provided base DNs can't be used to enable replication.
6	The provided base DNs can't be used to disable replication.
7	The provided base DNs can't be used to initialize the contents of the replicas.
8	Error connecting with the provided credentials.
9	Couldn't find the replication ID of the domain to be used to initialize the replica.
10	Exceeded the number of tries to start initialization. The server systematically receives a `peer not found` error.
11	Error enabling replication on a base DN.
12	Error initializing a base DN.
13	Error reading the configuration.
14	Error updating the topology registry.
15	Error reading the topology registry.
16	Error reading `TopologyCache`.

Exit code	Details
17	Error configuring the replication server.
19	Error disabling replication on a base DN.
20	Error removing the replication port reference on a base DN.
21	Error initializing the administration framework.
22	Error seeding the truststore.
23	Error launching pre-external initialization.
24	Error launching post-external initialization.
25	Error disabling a replication server.
27	The server must be offline to perform the operation.
28	Error updating the server configuration.
29	The server is unavailable (either can't be contacted or has an active unavailable alert).
30	The server joining the topology has the replication set defined, but no restricted domains have been selected.
31	Failed to lock the topology for a subcommand that requires no other commands running.
32	The location provided on the command line didn't match the location set in the server.
33	One or more replicas aren't connected to the replication topology when executing the `initialize-all` subcommand.
34	One or more replicas didn't complete the import in a timely fashion.
35	Either the first server, the second server, or both are running an older PingDirectory version than the one used for `dsreplication`.
36	Initialization didn't complete because it was canceled.
37	The `remove-defunct-server` subcommand was invoked for a still active server.
38	The `remove-defunct-server` subcommand was invoked with a defunct replication port that doesn't match the server's replication port in the topology registry.
39	The server is part of a mixed-version environment where some servers support the topology registry and others do not.
40	The replication subcommand is no longer supported.

Exit code	Details
41	Problem parsing the topology JSON file.
42	The internal seed server required to concurrently enable replication is unavailable.
43	The internal seed server required to enable replication is the same as the server added to the topology.
44	The base entry for the given base DN to replicate couldn't be added on both servers.
45	The server doesn't have mirrored subtrees, but the `dsreplication disable` command is being run from a server that does.
100	Other error.

Replication reference

This section covers general reference information about replication.

Summary of the dsreplication Subcommands

Use this topic as a reference for `dsreplication` subcommands and functions and their descriptions.

The following table presents a summary of the `dsreplication` subcommands and functions.

> **Tip**
>
> To see the available options for each subcommand, run `bin/dsreplication <subcommand> --help`.

Subcommand	Description
`disable`	Disables replication on the specified server for the provided base distinguish name (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 updates the configuration of all the servers. It is sufficient to execute the command line one time for each server you add to the replication topology.
`initialize`	Initializes the data under the specified base DN on the destination server with the contents on the source server.
`initialize-all`	Initializes the data under the specified base DN on all servers in the replication topology with the contents on the specified server.

Subcommand	Description
`post-external-initialization`	Used with `pre-external-initialization`, the command resets the generation ID based on the newly-loaded data. Consider the following using this subcommand: • You must call this subcommand 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. For more information, see the usage of the `pre-external-initialization` subcommand. • This subcommand only needs to be run one time on one of the replicas.
`pre-external-initialization`	Clears the existing generation ID and removes all accumulated changes of the replicated suffix from the Replication Changelog Database on every replication server. Consider the following using this subcommand: • Use this subcommand 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. • You only need to run this subcommand one time on one of the replicas.
`status`	Displays the status of replication domains. If no base DNs are specified as parameters, the information for all base DNs displays. 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 following is the distinguished name (DN) for the monitor entry.

```
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.
`generation-id`	Generation ID for the base DN on the PingDirectory 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 acknowledgment messages sent to this replica not currently used.
`ack-received`	Number of acknowledgment messages received from this replica not currently used.
`add-sent`	Number of protocol messages with an `LDAPAdd` sent to this replica.
`add-received`	Number of protocol messages with an `LDAPAdd` received from this replica.
`delete-sent`	Number of protocol messages with an `LDAPDelete` sent to this replica.
`delete-received`	Number of protocol messages with an `LDAPDelete` received from this replica.
`done-sent`	Number of done messages sent to this replica. A done message indicates an end of an 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.

Monitor Attribute	Description
`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. **Note** The number of messages should always be 0 because the replica never sends a heartbeat message to the server.
`initialize-request-sent`	Number of initialize-request messages sent to this replica. This message is sent when another replica requested initialization of its data using the replication protocol.
`initialize-request-received`	Number of initialize-request messages received from this replica. This message is sent when this replica requested initialization of its data using the replication protocol from another replica.
`initialize-target-sent`	Number of initialize-target messages sent to this replica. This message is sent before another replica has started the initialization of this replica.
`initialize-target-received`	Number of initialize-target messages received from this replica. This message is sent before this replica starts the initialization of one or more replicas.
`ldap-server:<hostname:ldapPort>`	The host name and port of the local replication server.
`modify-sent`	Number of protocol messages with an `LDAPModify` sent to this replica.
`modify-received`	Number of protocol messages with an `LDAPModify` received from this replica.
`modify-dn-sent`	Number of protocol messages with an `LDAPModify` DN sent to this replica.
`modify-dn-received`	Number of protocol messages with an `LDAPModify` DN received from this replica.
`repl-server-start-sent`	Number of replication-server-start messages sent to this replica. The Replication Server responds with this message to the start message received from the replica. **Note** The number of messages should never be more than 1.
`repl-server-start-received`	Number of replication-server-start messages received from this replica. **Note** The number of messages should always be 0.

Monitor Attribute	Description
`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. **Note** The number of messages should always be 0.
`server-start-sent`	Number of server-start messages sent to this replica. **Note** The number of messages should always be 0.
`server-start-received`	Number of server-start messages received from this replica. Server-start is the first message the replica sends after establishing a replication connection. **Note** The number of messages should never be more than 1.
`window-sent`	Number of window messages sent to this replica. Window messages are used for cumulative acknowledgment in the replication protocol.
`window-received`	Number of window messages received from this replica. Window messages are used for cumulative acknowledgment in the replication protocol.
`window-probe-sent`	Number of window probe messages sent to this replica. **Note** The number of messages should always be 0.
`window-probe-received`	Number of window probe messages received from this replica. The replica sends a window probe message to the server if the send window in the replica is closed and the replica is unable to publish updates to the server.
`update-sent`	Number of changes sent to this server.
`update-received`	Number of changes received from this server.
`internal-connection`	Indicates if the replica is in the same process as the replication server.
`server-state`	Displays the state of the replica. Displays the latest change number that the replica has seen from all the other replicas including itself.
`consumed-update-recent-rate`	Rate that the connected PingDirectory server is consuming updates sent by this replication server, expressed as the number of updates per second and measured over the last five seconds.

Monitor Attribute	Description
`consumed-update-peak-rate`	Highest rate that the connected PingDirectory server has consumed updates sent by this replication server, measured over a 5 second period from the start of the replication server.
`produced-update-recent-rate`	Rate that the connected PingDirectory server has sent updates to this replication server, expressed as the number of updates per second and measured over the last 5 seconds.
`produced-update-peak-rate`	Highest rate that the connected PingDirectory server has sent updates to this replication server, measured over a 5 second period from the start of the replication server.
`max-send-window`	Maximum number of changes that can send 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.
`degraded`	Indicates that the generation ID of the replica does not match the generation ID of the server. When loading data into the topology, this is a temporary state. If this is not a temporary state, the replica has not been initialized. **Note** The `degraded` value should be false.

Summary of the Indirect LDAP Server Monitor information

Use this topic as a reference for Indirect LDAP Server Monitor attribute information.

These attributes provide information about a PingDirectory server that is connected to a different replication server in the topology.

The following is the distinguished name (DN) for the Indirect LDAP server monitor entry.

```
dn: cn=Indirect LDAP Server <baseDN> <serverID>,cn=monitor
```

The following table provides a description of the attributes in the `cn=Indirect LDAP Server` monitor entry.

Monitor Attribute	Description
`replica-id: <serverID>`	ID number identifying the replica
`base-dn:<baseDN>`	Base DN

Monitor Attribute	Description
`connected-to: Remote Repl Server <baseDN> <host name:replPort> <repl ID>`	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 PingDirectory server and the most recent change seen by the remote replication server. **Note** This value is the amount of time between the time stamps, not the amount of time required to synchronize the 2 servers.
`generation-id`	Generation ID for this suffix on this remote replication server
`consumed-update-recent-rate`	Rate that the connected Replication Server is consuming updates sent by this replication server This value is expressed as the number of updates per second and measured over the last 5 seconds.
`consumed-update-peak-rate`	Highest rate that the connected Replication Server has consumed updates sent by this replication server This value is measured over a 5 second period beginning at the start of the replication server.

Summary of the Remote Replication Server Monitor information

Use this topic as a reference for Remote Replication Server Monitor attribute information.

The following is the distinguished name (DN) for the Remote Replication Server monitor entry.

```
dn: cn=Remote Repl Server <baseDN> <host name:replPort> <serverID>,cn=monitor
```

The following table provides a description of the attributes in the `cn=Remote Repl Server` monitor entry.

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`.

Monitor Attribute	Description
`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.
`restricted`	Indicates that remote replication server is in an entry-balancing deployment.
`add-sent`	Number of protocol messages with an `ldapadd` sent to the remote replication server.
`add-received`	Number of protocol messages with an `ldapadd` received from the remote replication server.
`delete-sent`	Number of protocol messages with an `ldapdelete` sent to the remote replication server.
`delete-received`	Number of protocol messages with an `ldapdelete` 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.

Monitor Attribute	Description
`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 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 `ldapmodify` sent to the remote replication server.
`modify-received`	Number of protocol messages with an `ldapmodify` received from the remote replication server.
`modify-dn-sent`	Number of protocol messages with an `ldapmodify` DN sent to the remote replication server.
`modify-dn-received`	Number of protocol messages with an `ldapmodify` DN received from the remote replication server.
`monitor-sent`	Number of monitor messages sent to the remote replication server. This message is used when communicating with PingDirectory servers running an earlier release.

Monitor Attribute	Description
`monitor-received`	Number of monitor messages received from the remote replication server. This message is primarily used when communicating with a PingDirectory server running an earlier release.
`monitor-request-sent`	Number of monitor requests sent to the remote replication server. The receiving server responds 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 responds 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. **Note** This monitor message is only used with PingDirectory server 3.5 or later.
`monitor-v2-received`	Number of monitor messages received from the remote server. **Note** This monitor message is only used with PingDirectory 3.5 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 stops sending update messages to the remote server when it receives this message.
`repl-server-start-sent`	Number of `replication-server-start` messages sent to the remote replication server. The Replication Server responds with this message to the `replication-server-start` message received from remote replication servers. **Note** The number of messages should never be more than 1.
`repl-server-start-received`	Number of `replication-server-start` messages received from the remote replication server. **Note** The number of messages should never be more than 1.
`reset-generation-id-sent`	Number of reset generation ID messages received from the remote replication server. This message is sent before and after the data is initialized in the topology.

Monitor Attribute	Description
`reset-generation-id-received`	Number of reset generation ID messages sent to the remote replication server. This message is sent before and after the data is initialized in the topology.
`server-info-sent`	Number of replication server information messages sent to the remote replication server. This message tells other replication servers about the replicas directly connected to the sending server. This message is also used to distribute information about the location of replicas.
`server-info-received`	Number of replication server information messages received from the remote replication server. This message tells other replication servers about the replicas directly connected to the sending server. This message is also used to distribute information about the location of replicas.
`set-source-location-sent`	Number of `set-source-locations` messages sent to the remote replication server. This message is used by WAN gateway servers to request update messages from additional locations.
`set-source-location-received`	Number of `set-source-locations` messages sent to the remote replication server. This message is used by WAN gateway servers to request update messages from additional locations.
`start-sending-updates-sent`	Number of `start-sending-updates` messages sent to the remote replication server. The remote server can 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 can only start after receiving this message.
`window-sent`	Number of window messages sent to the remote replication server. Window messages are used for cumulative acknowledgment in the replication protocol.
`window-received`	Number of window messages received from the remote replication server. Window messages are used for cumulative acknowledgment 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.

Monitor Attribute	Description
`consumed-update-recent-rate`	Rate that the connected PingDirectory server is consuming updates sent by this replication server, expressed as the number of updates per second and measured over the last 5 seconds.
`consumed-update-peak-rate`	Highest rate that the connected PingDirectory server has consumed updates sent by this replication server, measured over a 5 second period from the time the replication server was started.
`produced-update-recent-rate`	Rate that the connected PingDirectory server has sent updates to this replication server, expressed as the number of updates per second and measured over the last 5 seconds.
`produced-update-peak-rate`	Highest rate that the connected PingDirectory server has sent updates to this replication server, measured over a 5 second period since the replication server was started.
`max-send-window`	Maximum number of changes that can be sent to the remote replication server before requiring an ACK.
`current-send-window`	Number of changes remaining to be sent to the remote replication server before requiring an ACK.
`max-rcv-window`	Maximum number of changes that can be received from the remote replication server before sending an ACK.
`current-rcv-window`	Number of changes remaining to be received from the remote replication server before sending an ACK.
`degraded`	Indicates that the generation ID of the replica does not match the generation ID of the remote replication server. This is a temporary state when loading data into the topology or the replica has not been initialized. **Note** Normally, the `degraded` value 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.

The following is the distinguished name (DN) for the Replica Monitor server monitor entry.

```
dn: cn=Replica <baseDN>,cn=monitor
```

Monitor Attribute	Description
`base-DN: <baseDN>`	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 server has lost connection to a replication server
`received-updates`	Number of updates that the 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 because of an unresolvable naming conflict.
`replica-id`	Server ID for this replica.
`max-rcv-window`	Maximum number of changes that the 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 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.

Monitor Attribute	Description
`max-send-window`	Maximum number of changes that the 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 server.
`replication-backlog`	Number of changes that are from this replica.

Summary of the Replication Server Monitor information

Use this topic as a reference for the Replication Server Monitor attributes and information.

The following is the distinguish name (DN) for the replication server monitor entry.

```
dn: cn=Replication Server,cn=monitor
```

The following table provides a description of the attributes in the `cn=Replication Server` monitor entry.

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.

The following is the distinguished name (DN) for the Replica 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`. The following is an example entry. `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`. The following is an example entry. `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 directory information tree (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.

The following is the distinguished name (DN) for the Replica Server Database Monitor entry.

```
dn: cn=Replication Server Database Environment,cn=monitor
```

Monitor Attribute	Description
`je-version`	Current version of the Oracle Berkeley Java Edition.
`current-db-cache-size`	Current database (DB) cache size.
`max-db-cache-size`	Maximum DB cache size.
`db-cache-percent-full`	Percentage of the cache used by the 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.
`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.

Monitor Attribute	Description
`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.
`je-env-stat-lastCheckpointStart`	Location in the log of the last checkpoint start.
`je-env-stat-lockBytes`	Number of bytes of Java Edition (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 that were not in memory.
`je-env-stat-NCheckpoints`	Total number of checkpoints run so far.

Monitor Attribute	Description
`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 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.

Monitor Attribute	Description
`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.
`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 that 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.

Monitor Attribute	Description
`je-env-stat-NSequentialWriteBytes`	Number of bytes written which did not require repositioning the disk head more than 1MB from the previous file position.
`je-env-stat-NSequentialWrites`	Number of disk writes which did not require repositioning the disk head by more than 1MB from the previous file position.
`je-env-stat-NSharedCacheEnvironments`	Number of environments using the shared cache.
`je-env-stat-NTempBufferWrites`	Number of writes which had to be completed using the temporary marshalling buffer because the fixed size log buffers specified by `je.log.totalBufferBytes` and `je.log.numBuffers` were not large enough.
`je-env-stat-NToBe-CleanedLNsProcessed`	Accumulated number of LNs processed because they are soon to be cleaned.
`je-env-stat-NonEmptyBins`	Number of non-empty bins.
`je-env-stat-ProcessedBins`	Number of bins that were successfully processed by the INCompressor.
`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.

Monitor Attribute	Description
`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`.

Monitor Attribute	Description
`base-dn:<baseDN>`	Base DN summary.
`replica: replica-id, ldap-server connected-to, generation-id, replication-backlog, recent-update-rate, peak-update-rate, age-of-oldest- backlog-change`	Summary information for each replica in the topology. This entry appears for each replica in the topology with its own respective properties.
`replication-server: server-id, server, generation-id, status, last-connected, last-failed, failed-attempts, attributes.`	Summary information for each remote replication server only in the topology. This entry appears for each Replication Server in the topology with its own respective `serverID` and `server` properties.

Monitor Attribute	Description
`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, such as high load and network latency, then this queue fills up.When that happens, the sending replication server reads update messages from the changelog backend and 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 can be in the queue.This queue is capped by both the `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*** Set to 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`. **Tip** If this value changes frequently, the queue size setting is 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.

The following is the distinguished name (DN) for the replicationChanges Backend Monitor entry.

```
dn: cn=replicationChanges Backend,cn=monitor
```

Monitor Attribute	Description
`ds-backend-id`	ID descriptor for the backend. **Note** Typically, this is `replicationChanges`.
`ds-backend-base-dn`	Base DN for the backend. **Note** Typically, this is `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 following is the distinguished name (DN) for the 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. **Note** Initial buffer size is 4k.
`reallocations`	Number of times the buffers had to be reallocated because of insufficient size.
`large-buffer-creates`	Number of times a buffer larger than 512k was requested.

Monitor Attribute	Description
`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 section provides background reference information for advanced replication topics.

Topics in this section include:

- About the replication protocol
- Change number
- Conflict resolution
- WAN-friendly replication
- WAN Gateway Server
- WAN message routing
- WAN Gateway Server selection
- WAN replication in mixed-version environments
- Recovering a replication changelog
- Performing disaster recovery

About the replication protocol

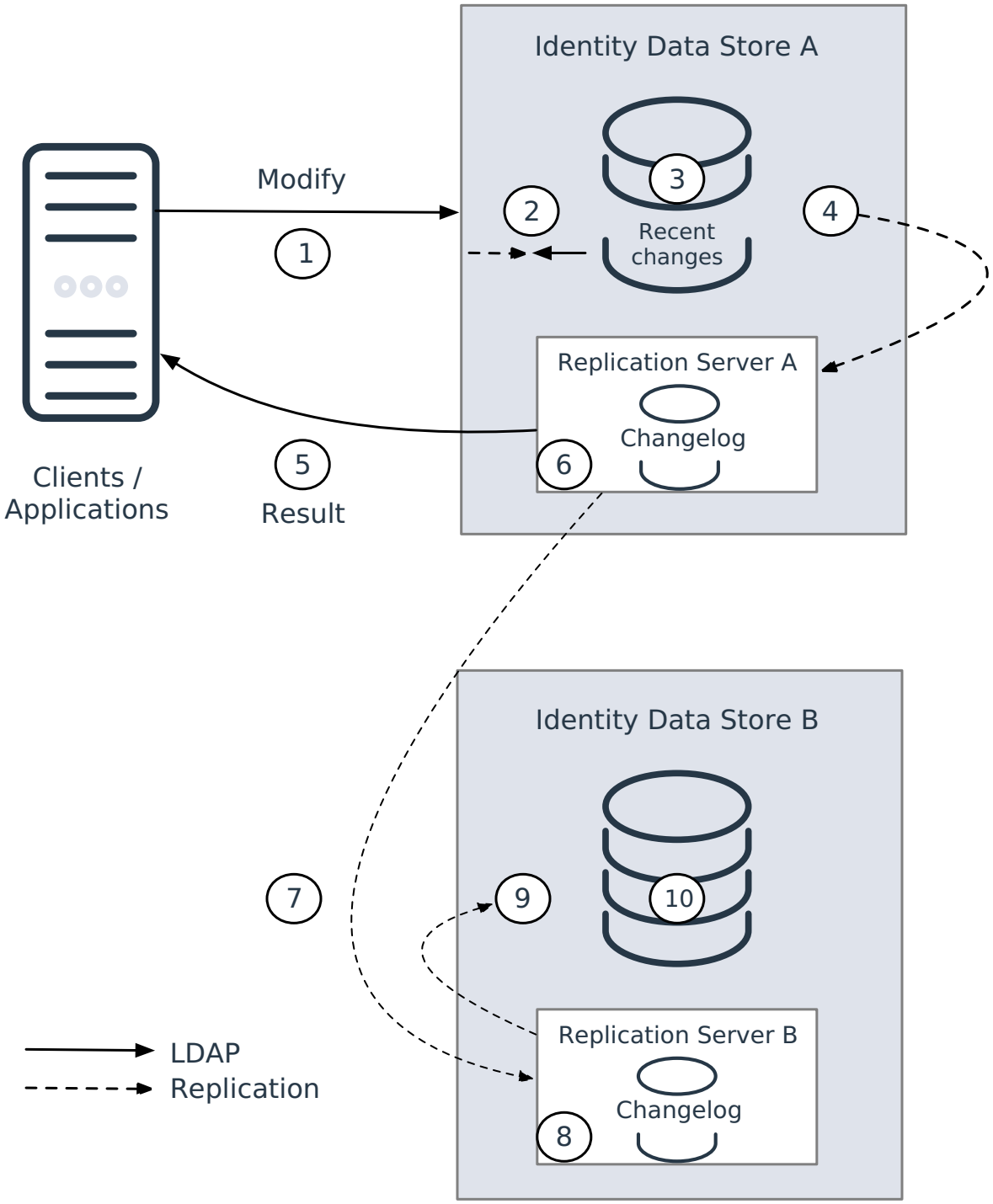
The topic covers the replication, topology, protocol process, and background information.

Replication communication uses a proprietary binary protocol that implements 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, and others are directed to all servers for monitoring requests.

In a replicated topology, each participating PingDirectory server connects to every other server through the replication server port to monitor health. Servers that share the same location setting are connected to rapidly replicate changes, and the WAN Gateway servers are all interconnected to replicate changes across locations.

The PingDirectory servers keep connections open as long as possible to reduce the communication latency when exchanging messages. Heartbeat messages transmit on a regular basis to detect a network failure or an unresponsive server as early as possible. Heartbeat messages also prevent idle connections from being closed by firewalls.

Replication Communication Flow



1. The client sends a `MODIFY` request to PingDirectory server A.

2. PingDirectory 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 server backend is updated to preserve the arrival order of client requests. Historical data in the target entry is updated to reflect the change.

> **Note**
>
> Historical data is only updated for `ADD` and `MODIFY` operations.

3. The PingDirectory server applies the modifications in the corresponding backend.
4. If the `MODIFY` operation successfully completes, then the PingDirectory server submits the update to its embedded replication server. The replication server is a component within the PingDirectory server process responsible for propagating updates to and from the replication topology. The PingDirectory server itself only communicates with a single replication server, whereas the replication server component is connected to all other replication servers. If the PingDirectory server process exits unexpectedly and some updates haven't 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 PingDirectory server and the replication server.
5. The response is sent to the client. In this example, a successful response is assumed.
6. The replication server records the update in its own Changelog backend, such as 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 PingDirectory server in the replication topology. When a server joins the replication topology after being disconnected for some reason, updates from the Replication Changelog backend are re-sent to this server. Old records from the Replication Changelog backend are purged, which, by default, removes records older than 24 hours. If the backend doesn't contain all of the records that another PingDirectory server needed when rejoining the replication topology, then the replicated data set in the PingDirectory server must be re-initialized. In this case, the server enters lockdown mode and sends an administrative alert.
7. The replication server submits the update to the replication server component in PingDirectory server B. If there were more PingDirectory servers in this example, the replication server would submit the update to all the other replication servers in the same location.
8. Just like in Step 6, the replication server component receiving an update inserts the change into its Replication Changelog backend.
9. The update forwards to the Replica in PingDirectory server B. Conflict resolution executes to see if the `MODIFY` request is in conflict with the existing attribute types or values in the existing entry.
10. The PingDirectory server applies the modification in the corresponding backend. The Recent Changes database doesn't update because only updates that originate at this server record in the Recent Changes database.

Change number

This topic provides the change number purpose, functions, and components.

As described in About the replication protocol, the PingDirectory server assigns a unique change number to each update operation, specifically `ADD`, `DELETE`, `MODIFY`, or `MODIFY DN` operations, to track each request received from a client. The change number is attached to each update propagated through replication and allows each PingDirectory server to order updates exactly the same way.

The change number is composed of the following fields:

`Timestamp`

Identifies when the update was made. The timestamp is time-zone-independent and recorded with millisecond resolution.

`Server ID`

Uniquely identifies the PingDirectory server where the update was made.

`Sequence number`

Defines the order of the updates received from external clients at a particular server..

> **Tip**
>
> The replication protocol sets a virtual clock that eliminates the need for synchronized time on servers. For troubleshooting purposes, you should keep the system clocks synchronized.

Conflict resolution

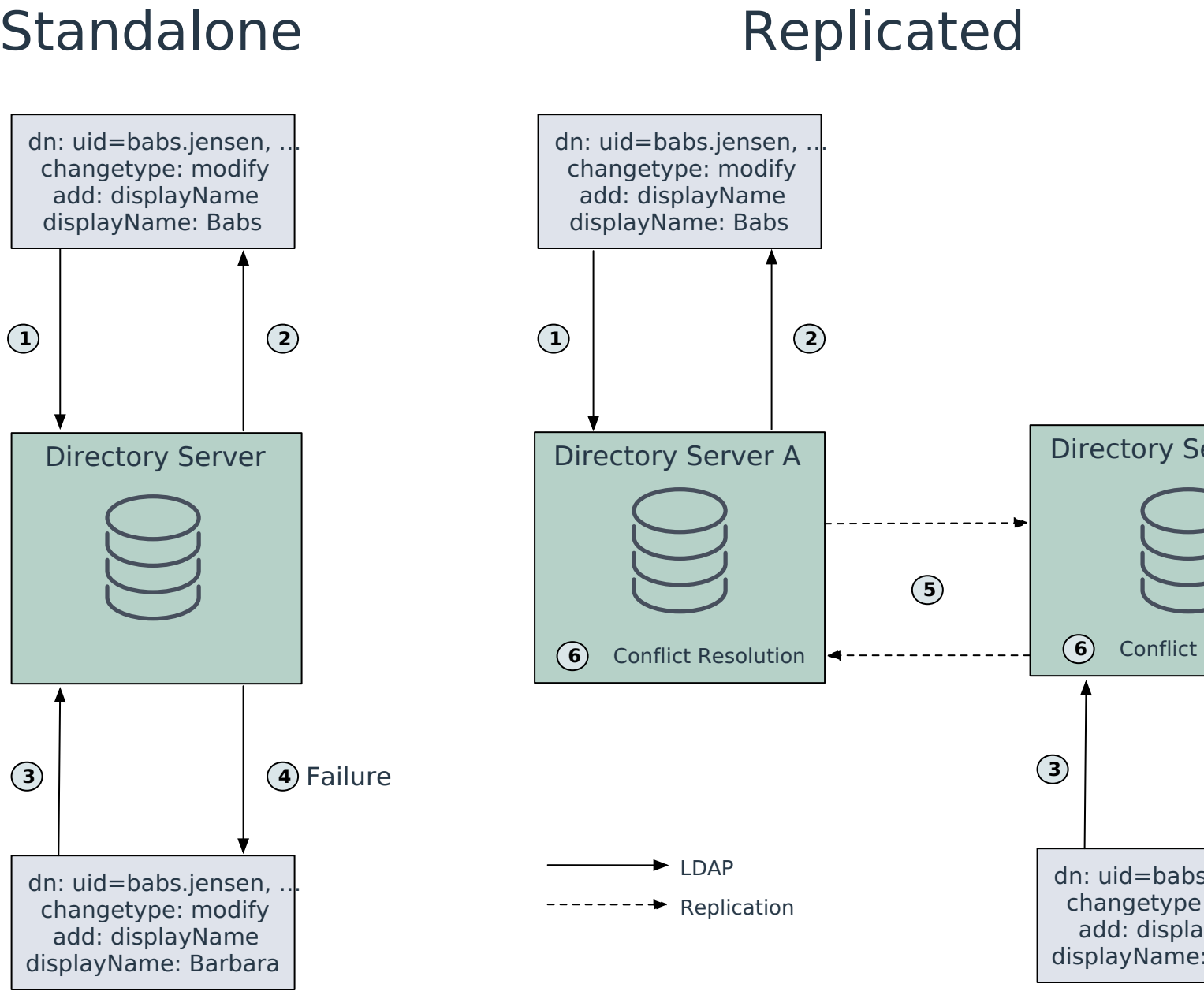
This topic covers background information, the conflict resolution process flow, and explanation of conflict resolution.

Replication employs an eventual-consistency model which can introduce a window where conflicting updates targeting the same entry might be applied at two different PingDirectory servers. In general, if the update that arrived later fails, two updates to the same server are in conflict. 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 simultaneously deleting the parent of this entry on another replica introduces 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 PingDirectory server uses a mechanism that orders all updates in the replication topology. The server assigns a unique change number to each update. The change number is attached to each update propagated through replication and allows each server to order updates exactly the same way. See the following diagram for an example.

Conflict resolution process flow

The following diagram depicts adding a single-valued attribute with different values to an entry concurrently at two PingDirectory servers resulting in a conflict as well as the conflict resolution in the standalone environment versus the replicated environment.



Processing steps

1. **The client sends a modify request to the Identity Datastore and succeeds.**
2. **The client sends another modify request that conflicts with the first.**
3. **In the standlaone environment, the request is rejected. In the replicated environment, the request succeeds.**
4. **In the replicated environment only: Replication propagates the updates.**
5. **In the replicated environment only: Conflict resolution concludes with A ignores the update from B, B sets the attribute value to the value on A.**

From the diagram example, you can make the following observations:

- The second operation fails if a client attempted to add the same attribute to the same entry at the same PingDirectory server.
- In a replicated environment, the conflict is not immediately seen if these updates are applied concurrently at two different PingDirectory servers.
- The conflict is handled only after replication propagates the updates.
- The PingDirectory servers resolve the conflict independently of the other server. On one server, the entry updates to reflect the correct value. On the other server, the value stays the same. As a result, each server independently resolves the conflict the same way based on the ordering of the updates.

WAN-friendly replication

Minimize latency with the PingDirectory server if you are using a WAN for your client applications.

Many multinational 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 PingDirectory server assigns one of following two roles to the replication servers:

- The role of standard replication transmitting updates to the other co-located replication servers.
- The role of 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.

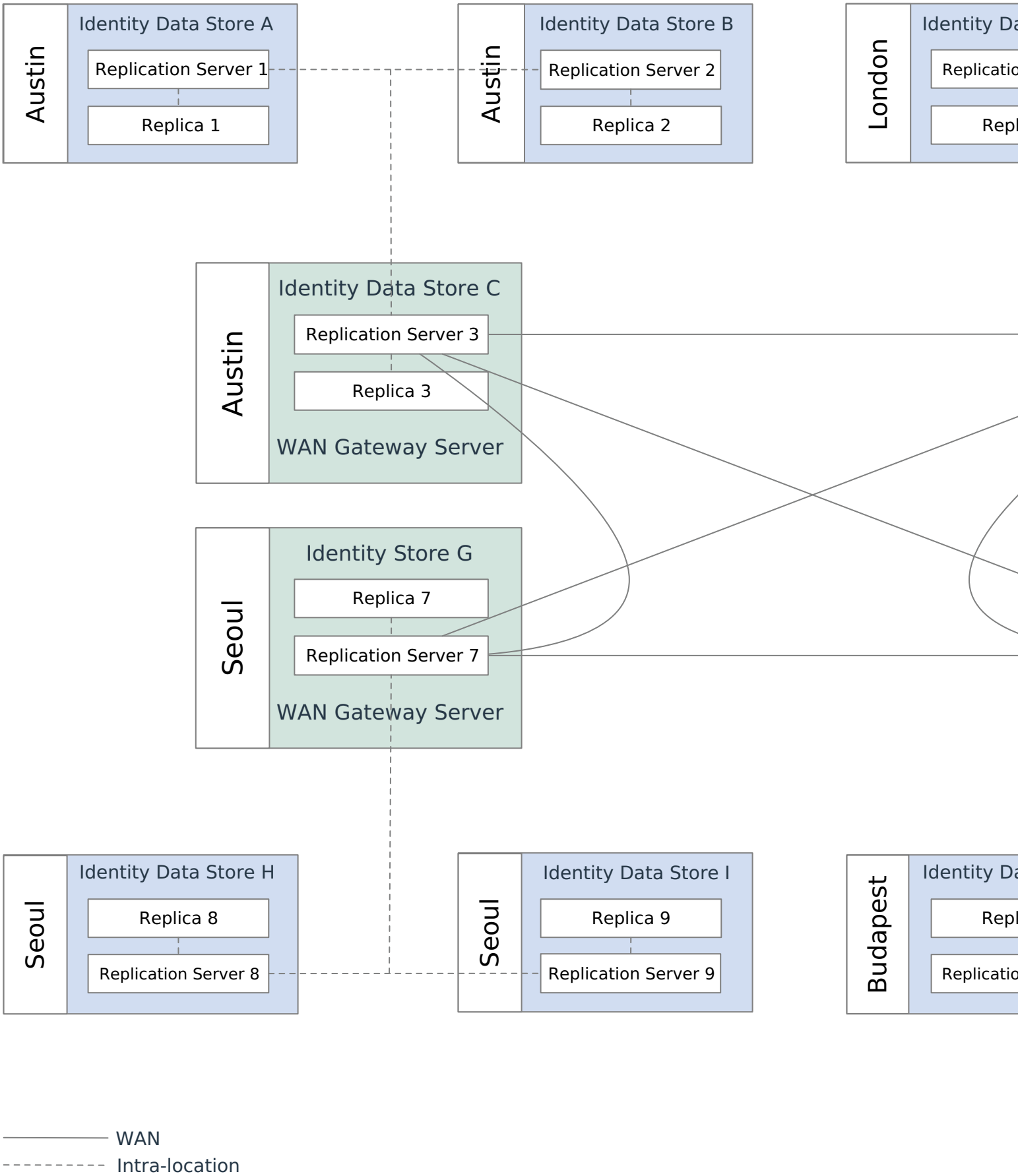
WAN Gateway Server

The PingDirectory 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. At the destination point, the replication server with the WAN Gateway role receives update messages from other WAN gateway servers at other locations and pushes them out to all replication servers at the current location.

This setup ensures that all WAN communication flows through the WAN Gateway Servers.

The following diagram shows a basic connection configuration for updates. All of the replication servers are fully connected to each other for monitoring or server negotiation purposes.



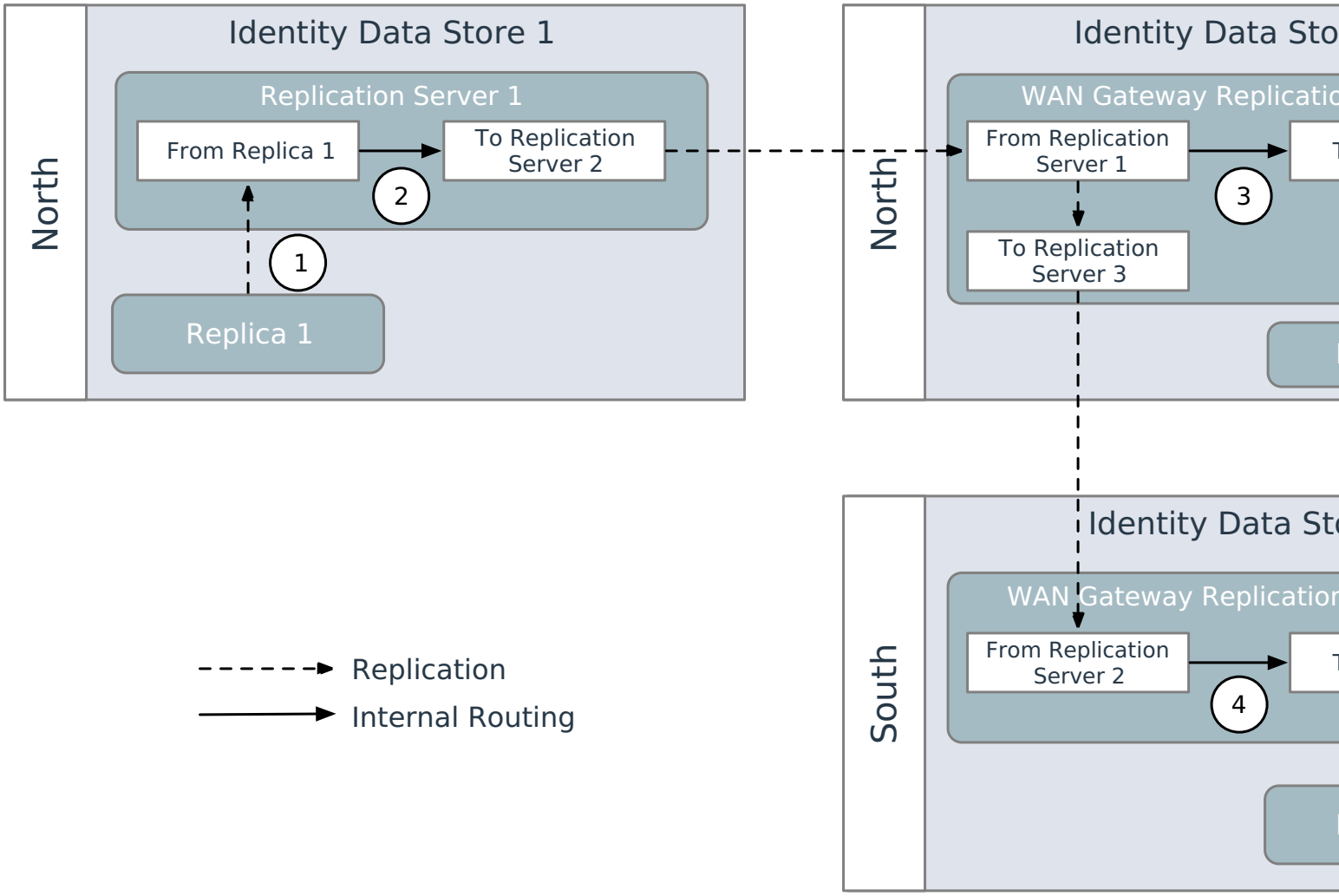
If the WAN Gateway Server is temporarily unavailable because of planned or unplanned downtime, the system dynamically re-routes updates to a newly-designated WAN Gateway Server in the same location. The replication server with the next highest WAN Gateway priority number automatically assumes the WAN Gateway role. For deployments with entry-balancing Directory Proxy Servers, there is one WAN Gateway Server per data set.

> **Note**
>
> By default, all servers are enabled to serve as WAN Gateways and all are set to priority 5, which makes them all equal. After replication has been enabled, you can change the WAN Gateway priority using `dsconfig`.

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 following figure illustrates how an update message is routed from a non-gateway server to a remote location.



Processing steps

1. Update received at Identity Datastore 1 at the north location is sent to the embedded Replication Server.

2. Replication Server 1 routes message to the WAN Gateway only.
3. The WAN Gateway at the North location (Identity Data Store 2) routes message to Replica 2 and to the WAN Gateway at the south location.
4. Replication Server 3, which is the WAN Gateway at the south location, routes the message to Replica 3.

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 (priority 1 is the highest priority) assumes this role.

A replication server does 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 gives 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, such as to become a WAN Gateway, if necessary, without any manual administrative action.

WAN replication in mixed-version environments

This topic covers the WAN Gateway role, replication actions, and updates in PingDirectory server mixed version environments.

Consider the following when using the WAN Gateway role in mixed-version environments:

- WAN Gateway Role assignment is only available on PingDirectory servers version 3.5 and later.
- Legacy servers earlier than 3.5, are never selected as WAN Gateways.
- Updates from legacy replication servers forward to all replication servers and do not funnel to a single WAN Gateway, which increases WAN during replication.
- Updates from remote WAN Gateways forward to the legacy replication servers, which increases WAN traffic during replication.

Recovering a replication changelog

If the replication changelog, `<server-root>/changelogDb`, is compromised because of a disk or NAS failure, you can recover a replication changelog.

About this task

To recover a replication changelog:

Steps

1. Stop the server.
2. Backup `replicationChanges` from a remote server using 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 using the following command.

```
$ bin/restore --backupDirectory /app/tmp/replicationChanges
```

4. Start the server.

Performing disaster recovery

If data is compromised across all systems and a restore is necessary, you can perform disaster recovery.

About this task

Note

Consider the following 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 7 days of exports are maintained.

  Note

  You should archive these exports 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, such as if an errant application mistakenly wipes out some data.

For more information about LDIF exports, see LDIF export as a recurring task.

> **Note**
>
> These steps assume that no server performs read or write operations during this process.

To complete a restore of your systems and perform a disaster recovery:

Steps

1. Stop all servers.
2. Go to one of the servers and remove it from the topology:

   Example:

   ```
   bin/remove-defunct-server --performLocalCleanup --no-prompt
   ```

3. Locate the backup or exported LDIF file that represents the last working copy of the database.
4. Restore the backup or import the LDIF file on a single server. If importing an LDIF file, use the `--excludeReplication` option with the `bin/import-ldif` command.
5. Start the restored server. The server can now receive client requests.
6. Clean up replication artifacts from the next server before starting it up:

   Example:

   ```
   bin/remove-defunct-server --performLocalCleanup --no-prompt
   ```

7. Start the server in lockdown mode with the following command:

   Example:

   ```
   bin/start-server --skipPrime --lockdownMode
   ```

8. Enable replication from the first server to the second server.

   Example:

   ```
   bin/dsreplication enable
   ```

9. Initialize the second server from the first with the following command:

   Example:

   ```
   bin/dsreplication initialize
   ```

10. Restart the second server or use the `bin/leave-lockdown-mode` command to exit lockdown mode.

    The second server can now receive client requests.

11. Repeat steps 5 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 and configure custom log publishers with pre-defined filtering with its own log rotation and retention policies.

Default PingDirectory server logs

The PingDirectory server provides a standard set of default log files to monitor the server activity.

PingDirectory server logs

View this set of logs in the `PingDirectory/logs` directory. The following default log files are available on the PingDirectory server and are defined in the following table.

Log File	Description
`access`	File-based Access Log that records operations processed by the PingDirectory server. Access log records can be used to provide information about problems during operation processing, such as unindexed searches, failed requests, and so forth, 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 PingDirectory server configuration in a format that can be replayed using the `dsconfig` tool.
`errors`	File-based Error Log. Provides information about warnings, errors, and significant events that occur during server processing.
`expensive-ops`	Expensive Operations Log. Disabled by default. Provides only those operations that took longer than 1000 milliseconds to complete.
`failed-ops`	Failed Operations Log. Provides information on all operations that failed in single-line log format.
`replication`	Records any replication errors and publishes them to the 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.

Log File	Description
`server.out`	Records anything written to standard output or standard error, which includes startup messages. If garbage collection debugging is enabled, then the information is 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 PingDirectory server with the `setup` tool.
`tools`	Directory that holds logs for long running utilities `import-ldif`, `export-ldif`, `backup`, `restore`, and `verify-index`. Current and previous copies of the log are present in the PingDirectory server. Also contains `dsreplication` subcommand logging.
`update.log`	Records messages that occur during a PingDirectory 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 log publishers
- Debug log publishers
- Error log publishers

Each type has multiple subtypes of log files based on the log server type.

Access log publishers

Access log publishers provide the requests received and the responses returned by the PingDirectory server. This 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.

For information on tools that can analyze the access log to provide summaries of LDAP activity and performance, see Monitoring PingDirectory metrics with Splunk and the `summarize-access-log` tool description in Available command-line tools.

Access log type	Description
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. You can find this log publisher in the `<pd_install>/logs/data-recovery/data-recovery` directory.
JDBC-based access log	Stores access log information using a Java Database Connectivity (JDBC) database connection. Disabled by default.

Access log type	Description
File-based access logs	Provides a character-based stream used by TextWriter publishers as a target for outputting log records. There are several types of file-based access 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. You can find this log publisher in the `<pd_install>/logs/access` directory. ***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. You can find this log publisher in the `<pd_install>/logs/expensive-ops` directory. ***Failed-Operations Access Log*** Publishes only those access log messages of operations that failed for any reason. Enabled by default. You can find this log publisher in the `<pd_install>/logs/failed-ops` directory. ***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. You can find this log publisher in the `<pd_install>/logs/access.json` directory. ***Console JSON Access Logger*** Publishes JSON-formatted access log messages to the Java virtual machine (JVM)'s standard output or standard error stream. **Note** 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.

Access log type	Description
HTTP operation log publishers	*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. You can find this log publisher in the `<pd_install>/logs/http-detailed-access` directory.

Debug log publishers

Debug log publishers provide information about warnings, errors, or significant events that occur within the server.

Debug log type	Description
Debug loggers	*Debug ACI Logger* Stores debug information on access control instruction (ACI) evaluation for any request operations against the server. You can find this log publisher in the `<pd_install>/logs/debug-aci` directory. *Server SDK Extension Debug Logger* Provides simplified access to debug messages generated by Server SDK extension. You can find this log publisher in the `<pd_install>/logs/server-sdk-extension-debug` directory.

Error log publishers

The Error log reports errors, warnings, and informational messages about events that occur during the course of the server's operation.

Error log type	Description
File-based error logs	There are several types of File-based error logs: ***Error log*** Publishes error messages to the file system. Enabled by default. You can find this log publisher in the `<pd_install>/logs/errors` directory. ***Replication log*** Publishes replication error messages to the file system. Enabled by default. You can find this log publisher in the `<pd_install>/logs/replication` directory. ***JSON Error Logger*** Publishes JSON-formatted error log messages to the file system. You can find this log publisher in the `<pd_install>/logs/errors.json` directory. ***Console JSON Access Logger*** Publishes JSON-formatted access log messages to the JVM's standard output or standard error stream. **Note** 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 syslog port.

Viewing the list of log publishers

View the list of log publishers on the PingDirectory server using the `dsconfig` tool.

About this task

Note

You must specifically configure the Java Database Connectivity (JDBC), syslog, and admin alert log publishers using `dsconfig` before they appear in the list of log publishers.

Steps

- To view the log publishers, run `dsconfig` with the `list-log-publishers` subcommand.

Example:

```
$ bin/dsconfig list-log-publishers
```

Result:

```
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 PingDirectory server using the `dsconfig` tool.

About this task

By default, the following loggers are disabled and should only be enabled 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

Steps

- To enable an access log publisher, run `dsconfig` with the `set-log-publisher-prop` subcommand.

  Example:

  In this example, the Expensive-Operations Access Logger is enabled, which records 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 PingDirectory 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 PingDirectory 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 show only specific information necessary for troubleshooting purposes.

Admin-Alert Access Logs

Disabled by default. The Admin-Alert Access Log is a 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 PingDirectory 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 Java Database Connectivity (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 PingDirectory 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, such as 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 PingDirectory 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 and 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 PingDirectory server.

By default, each operation generates one access log message.

The access log displays the following common properties.

Property	Description
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` • `DISCONNECT` • `SEARCH REQUEST/RESULT` • `MODIFY REQUEST/RESPONSE` • `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 are re-used when the server restarts or when it has had enough connections to cause the identifier to wrap back to zero.

Property	Description
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 are 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 succeeded or 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 PingDirectory 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 PingDirectory 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, that 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] DISCONNECT conn=0 reason="Client Unbind"
... (more output) ...
```

Modifying the access log using dsconfig interactive mode

The File-Based Access Log can be modified to include or exclude all log messages of a given type using the `dsconfig` tool in interactive or non-interactive mode.

About this task
Steps

1. Use `dsconfig` in interactive mode to modify the access log properties.

   Example:

   ```
   $ 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 PingDirectory 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, enter the number corresponding to the property that you want to change, and then follow the prompts.
7. Type `f` to apply the changes.

dsconfig non-interactive mode">

Modifying the access log using `dsconfig` non-interactive mode

Use the `dsconfig` tool in non-interactive mode to modify a log publisher property on the command line or in a script.

About this task

For information on each property, see the PingDirectory server Configuration Reference Guide, which is an HTML document listing the various properties for each PingDirectory server component.

Steps

- Use `dsconfig` with the `--no-prompt` option with the properties you want to modify or set for your access log.

  Example:

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
```

Adding connection information to request-type log messages

You can add connection information to request-type log messages for file-based access log publishers of type `file-based-access` or `json-access`.

About this task

If your applications use the outputs of access log publishers to trigger workflow responses, you can enrich those logs with connection information related to bind, search, and modify access requests. To add this connection information, enable the `include-connection-details-in-request-messages` configuration property for existing and new file-based access log publishers.

> **Note**
>
> The `include-connection-details-in-request-messages` property is disabled by default. The property supports only `file-based-access` and `json-access` type file-based access log publishers.

When you enable the `include-connection-details-in-request-messages` property, the log messages for request-based connection types include the following information:

- Client IP address
- Client port (LDAP connections only)
- PingDirectory server IP address
- PingDirectory server port (LDAP connections only)
- Communication protocol

Steps

- Enable the `include-connection-details-in-request-messages` property for a file-based access log publisher of type `file-based-access` or `json-access`.

  Choose from:

  - Enable the property in an existing log publisher. For example:

    ```
    $ bin/dsconfig set-log-publisher-prop \
      --publisher-name "Test JSON Access Log Publisher" \
      --set include-connection-details-in-request-messages:true
    ```

> **Note**
>
> You can disable `include-connection-details-in-request-messages` by setting it to `false`.

- Create a new log publisher with the property enabled. For example:

```
$ bin/dsconfig create-log-publisher \
  --publisher-name "Test Writer Access Log Publisher" \
  --type file-based-access \
  --set enabled:true \
  --set log-connects:true \
  --set log-disconnects:true \
  --set log-security-negotiation:true \
  --set log-requests:true \
  --set log-file:logs/testaccess.log \
  --set include-extended-search-request-details:true \
  --set include-requester-ip-address:true \
  --set include-requester-dn:true \
  --set include-request-controls:true \
  --set include-response-controls: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 include-connection-details-in-request-messages:true
```

Modifying the maximum length of log message strings

Change the maximum length of log message strings by setting the `max-string-length` configuration property.

About this task

By default, the PingDirectory 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. If any string has more than the configured number of characters, then that string is truncated and a placeholder is appended to indicate the number of remaining characters in the original string.

Steps

- To set the `max-string-length` property for an access log, run `dsconfig`.

  Example:

  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

Disable logging of inter-server periodic search requests.

Steps

1. Create a request criteria that matches all search requests with the inter-server request control.

   Example:

   ```
   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.

   Example:

   ```
   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 disables the logging of internal search requests and their corresponding results to the access log.

   Example:

   ```
   dsconfig set-log-publisher-prop --publisher-name "File-Based Access Logger" \
           --set "request-criteria:Exclude Inter-server Searches"
   ```

Generating access log summaries

The PingDirectory server provides a convenience tool, `summarize-access-log`, that generates a summary of one or more file-based access logs.

About this task

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 uses when processing the filter)

  These filters are represented in a generic manner so that any distinct assertion values or substring assertion elements are replaced with question marks and attribute names in all lowercase characters such as `(givenName=John)` would be represented as `(givenName=?)`.

Steps

- Use the **`bin/summarize-access-log`** with the path to one or more access log files.

  Example:

  ```
  $ bin/summarize-access-log /path/to/logs/access
  ```

  Result:

  The system displays the following information.

```
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%)
```

Excluding specific log messages

To make log files smaller and easier to analyze, you can create a log publisher message exclusion policy.

About this task

Log publisher message exclusion policies suppress messages from being logged to specific log publishers. You can create a policy using `dsconfig create-log-publisher-message-exclusion-policy` and then assign it to an existing log publisher.

Exclusion policies currently only support error log publishers. You can define exclusion policies for PingDirectory, PingDirectoryProxy, and PingDataSync.

> **Warning**
>
> Enabling log message exclusion could cause the server to fail to log important messages. Be as specific as possible when configuring exclusion policies to ensure that you are only suppressing the intended log messages.

The following table describes the attributes and arguments for creating a log publisher message exclusion policy with `dsconfig create-log-publisher-message-exclusion-policy`:

Attribute or argument	Description	Accepted values	Requirement
`--type`	Specifies the type of log publisher to which the policy applies.	`error`	Required
`--policy-name`	Defines the name of the log message exclusion policy.	n/a	Required

Attribute or argument	Description	Accepted values	Requirement
`log-message-category`	Specifies the category of the log message to exclude.	`CORE, EXTENSIONS, PROTOCOL, CONFIG, LOG, UTIL, SCHEMA, PLUGIN, JEB, BACKEND, TOOLS, TASK, ACCESS_CONTROL, ADMIN, REPLICATION, VERSION, QUICKSETUP, ADMIN_TOOL, DSCONFIG, USER_DEFINED, EXAMINER, SYNC, PROXY, UPDATE, CONSOLE, CONSOLE_TOOLS, COMMON, METRIC, ID_SERVER, MONITORING, AUTHORIZATION, THIRD_PARTY, COMPONENTS, RUNTIME_INFORMATION`	Required
`log-message-severity`	Specifies the severity of the log message to exclude.	`FATAL_ERROR, INFORMATION, MILD_ERROR, MILD_WARNING, NOTICE, SEVERE_ERROR, SEVERE_WARNING, DEBUG`	Required
`log-message-regex`	Defines the regex string that must match the actual message of the log.	Any valid regex pattern.	Required
`enabled`	Specifies whether or not the policy suppresses log messages.	A value of `true` activates the log message suppression. A value of `false` deactivates the suppression.	Required
`log-message-id`	Specifies the ID of the log message to suppress.	Any positive integer value. If the message ID isn't declared, then the policy checks for a message ID value of `-1`.	Optional

Steps

1. Create a log publisher message exclusion policy.

   Example:

```
$ bin/dsconfig create-log-publisher-message-exclusion-policy \
  --type error \
  --policy-name "Test Policy" \
  --set "enabled:true" \
  --set "log-message-category:LOG" \
  --set "log-message-severity:NOTICE" \
  --set "log-message-regex:foobar" \
  --set "log-message-id:1880555611"
```

> **Note**
>
> To exclude a log message using this policy, all the attributes of the policy must match the fields of the log message. If they don't, the log message gets published.

2. Assign the exclusion policy to an existing log publisher using the `log-message-exclusion-policy` attribute.

   Example:

```
$ bin/dsconfig set-log-publisher-prop \
  --publisher-name "Example Error Log Publisher" \
  --set "log-message-exclusion-policy:Test Policy"
```

About log compression

The 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 when the logger is configured. Because of problems in trying to append to an existing compressed file, if the server encounters an existing log file at startup, it rotates 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 might want to have a second logger defined that does not use compression and has rotation and retention policies that minimizes 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.

Configure compression by setting the `compression-mechanism` property to have the value of `gzip` when creating a new logger.

About log signing

The 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 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.

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 is not completely verifiable because it still contains unsigned content from before signing was enabled. Only log files whose entire content was written with signing enabled are considered completely valid. For the same reason, if a log file is still open for writing, then signature validation does not indicate that the log is completely valid because the log doesn't 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.

After 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 server lets you encrypt log files as they are written.

The `encrypt-log` configuration property controls whether encryption is enabled for the logger. Enabling encryption causes the log file to have an `.encrypted` extension. If both encryption and compression are enabled, the extension is `.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 after 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 reduces 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 is performed with the server's preferred encryption settings definition.

To explicitly specify which definition should be used for the encryption, set the `encryption-settings-definition-id` property with the ID of that definition. You should set the encryption settings definition to be 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. You can also create a randomly generated encryption settings definition, 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 might 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

Configure log signing for a log publisher.

Steps

1. To enable log signing for a log publisher, use `dsconfig`.

   Example:

   In this example, the `sign-log` property is set 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.

   Example:

```
$ 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

The 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.

  Example:

  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
  ```

  Result:

  ```
  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

Configure log file encryption for a log publisher.

Steps

1. To enable encryption for a log publisher, use `dsconfig`.

   Example:

   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. Decrypt and decompress the file.

Example:

```
$ 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/Data-Sync/decrypted-access' using a
key generated from encryption settings definition
'332c846ef0dcd1d5187c1592e4c74cad33fc1e5fc20b726cd301cdd2b3ffbc2b'
Successfully wrote 123,456,789 bytes of decrypted data
```

Log sanitization

The PingDirectory server allows you to sanitize information as it's written to the access log so that you can prevent the system from logging sensitive information.

> **Note**
>
> To learn about the mechanisms to protect information after it has already been logged, see Sanitizing log files.

The PingDirectory server can sanitize log content on a field-by-field basis for default text-based (`name=value`) and JSON-formatted access logs. You can also use log sanitization for any destination where these messages are written, such as log files, syslog, standard output, and standard error messages.

You can control log content for all access logs or on a per-log or per-field basis, including fields generated by third-party extensions. You can also specify a default behavior for all fields of a specified type, such as applying a default sanitization type for all distinguished names (DNs) and search filters.

Configuration elements

There are three main configuration elements for customizing log sanitization:

Log field syntaxes

These define the default behavior for each syntax and can specify additional configuration for these syntaxes (for example, the included/excluded LDAP attributes for distinguished names (DNs) and filters or fields for JSON objects).

For more information, see Customizing log field syntaxes.

Log field behaviors

These can be used to define specific behaviors on a per-field basis and an optional overall default behavior for fields that are not explicitly configured.

For more information, see Customizing log field behaviors.

Access loggers

These can be associated with log field behaviors or can default to the log field syntax configuration.

For more information on the description of the behavior of each log sanitization option, see Log sanitization options.

Log sanitization options

There are six options you can use for sanitizing access log fields as they are being written.

Option	Description
`Preserve`	Preserves the original value without attempting to obscure it. **Note** Some sanitization might be performed, such as truncating very long values and escaping special characters, but this is more for usability than privacy.
`Omit`	Excludes the field completely from the log. Neither the field name nor its value will be present in the log.

Option	Description
`Redact Entire Value`	**Includes the field name in log messages, but replaces the entire value with a fixed string that does not reveal anything about the actual value for the field.** **To attempt to preserve the original syntax of a field value, the actual string that is used for the redacted value of a field depends on the syntax of that field. The redacted strings include:** ***String, string list, and Boolean fields*** **These fields will have a redacted value of `{REDACTED}`.** **Note** Because Boolean values can only be true or false, redacted and tokenized representations of Boolean values will not conform to the Boolean syntax. If there is concern about leaking sensitive information through a Boolean field, you should omit the field. ***Integer fields*** **These fields will have a redacted value of `-999999999999999999`.** ***Floating-point fields*** **These fields will have a redacted value of `-999999.999999`.** ***Distinguished name (DN) fields*** **These fields will have a redacted value of `redacted={REDACTED}`.** ***Filter fields*** **These fields will have a redacted value of `(redacted={REDACTED})`.** ***JSON object fields*** **These fields will have a redacted value of `\{ "redacted":"{REDACTED}" }`.** ***Generalized time fields*** **These fields will have a redacted value of `99990101000000.000Z`.** ***RFC 3339 timestamp fields*** **These fields will have a redacted value of `9999-01-01T00:00:00.000Z`.**

`Redact Value Components`	**Includes the field name in log messages but replaces the value with a string that can redact only certain components of the value without redacting the entire value.** **This preserves some usability of the original value without exposing sensitive information.** **Syntaxes that support redacted value components include:** ***String list*** **Each element in the list will be replaced with `{REDACTED}`, so a list with three elements will be represented as `{REDACTED},{REDACTED},{REDACTED}`.** ***DN*** **Attribute names will be preserved, but attribute values will be redacted. For example, the DN `uid=jdoe,ou=People,dc=example,dc=com` will be redacted as `uid={REDACTED},ou={REDACTED},dc={REDACTED},dc={REDACTED}`.** ***Filter*** **As with DNs, attribute names will be preserved, but attribute values will be redacted. For example, the filter `(&(givenName=John)(sn=Doe))` would be redacted as `(&(uid={REDACTED})(sn={REDACTED}))`.** **Note** You can configure the DN and filter log field syntaxes to include or exclude specified attributes from redaction while preserving the other attribute values: • To specify a set of included sensitive attributes so that only those attributes will have their values redacted, and to preserve all other attribute values, use the `included-sensitive-attribute` property. • To specify a set of excluded sensitive attributes so that only the values of the excluded attributes will be preserved, and to redact all other attribute values, use the `excluded-sensitive-attribute` property. ***JSON objects*** **Field names will be preserved but field values will be replaced with the string `{REDACTED}`.**

Option	Description
	Note You can configure the JSON log field syntax to include or exclude specified fields from redaction while preserving the other field values: • To specify a set of included sensitive fields so that only those fields will have their values redacted, and to preserve all other field values, use the `included-sensitive-field` property. • To specify a set of excluded sensitive fields so that only the values of the excluded fields will be preserved, and to redact the values of all other fields, use the `excluded-sensitive-field` property. **If a syntax does not have a value with components, the entire value is redacted.**

Option	Description
`Tokenize Entire Value`	Includes the field name in log messages but replaces the entire value with a tokenized representation that's based on the original value but doesn't reveal the value. This is similar to redacting the entire value, but because the tokenized value is based on the original value, and the logic used to perform the tokenization is repeatable, you can identify cases in which the same value appears multiple times in the log even if you do not know the original value. • String, string list, and Boolean fields will have a tokenized value of `\{TOKENIZED:` *<token-value>* `}`, where *<token-value>* is the actual value for the token generated from the original value (for example, `\{TOKENIZED:NcMNHJxmCwhETFIe}`). • Integer fields will have a tokenized value that starts with `-999999999` and is followed by nine more digits generated from the original value (for example, `-999999999600205901`). • Floating-point fields will have a tokenized value that starts with `-999999`. and is followed by six digits generated from the original value (for example, `-999999.695201`). • DN fields will have a tokenized value of `tokenized=\{TOKENIZED:` *<token-value>* `}`, where *<token-value>* is the value generated from the normalized string representation of the original DN. • Filter fields will have a tokenized value of `(tokenized=\{TOKENIZED:` *<token-value>* `})`, where *<token-value>* is the value generated from the normalized string representation of the original filter. • JSON object fields will have a tokenized value of `\{ "tokenized":"\{TOKENIZED:` *<token-value>* `}" }`, where *<token-value>* is the value generated from the normalized string representation of the original JSON object. • Generalized time and RFC 3339 timestamp fields will have a tokenized value with a year of `8888`, with the remainder of the month, day, hour, minute, second, and sub-second elements generated from the original timestamp.

`Tokenize Value Components`	**Behaves in the same way as redacted value components, but replaces the redacted string with a tokenized string. Syntaxes that support tokenized value components include:** ***String list*** **Each element in the list will be tokenized individually.** ***DN*** **Attribute names will be preserved, but attribute values will be tokenized.** ***Filter*** **Attribute names will be preserved, but attribute values will be tokenized.** **Note** You can configure the DN and filter log field syntaxes to include or exclude specified attributes from tokenization while preserving the other attribute values: • To specify a set of included sensitive attributes so that only those attributes will have their values tokenized, and to preserve all other attribute values, use the `included-sensitive-attribute` property. • To specify a set of excluded sensitive attributes so that only the values of the excluded attributes will be preserved, and to tokenize all other attribute values, use the `excluded-sensitive-attribute` property. ***JSON objects*** **Field names will be preserved, but field values will be tokenized.**

Option	Description
	Note You can configure the JSON log field syntax to include or exclude specified fields from tokenization while preserving the other field values: • To specify a set of included sensitive fields so that only those fields will have their values tokenized, and to preserve all other field values, use the `included-sensitive-field` property. • To specify a set of excluded sensitive fields so that only the values of the excluded fields will be preserved, and to tokenize the values of all other fields, use the `excluded-sensitive-field` property.

After you have identified the approach you want to take for sanitizing your log content as it is written, see Customizing log field syntaxes or Customizing log field behaviors.

Customizing log field syntaxes

Use customized, syntax-based log sanitization to make results as useful as possible while preserving the privacy of sensitive content.

About this task

The following log field syntaxes are supported by the log sanitization functionality:

- **String**
- **String list**
- **Boolean**
- **Integer**
- **Floating-point number**
- **Distinguished name (DN)**
- **LDAP search filter**
- **JSON object**
- **Generalized time timestamp**
- **RFC 3339 timestamp**

The following values are accepted for the `default-behavior` property:

- `preserve`
- `omit`
- `redact-entire-value`
- `redact-value-components`
- `tokenize-entire-value`
- `tokenize-value-components`

> **Note**
>
> For more information on the behavior of each value, see Log sanitization options.

Steps

1. To update the default configuration for log field syntax behavior, run `dsconfig set-log-field-syntax-prop`.

   Example:

   The following example updates the DN syntax to indicate that DNs should use component-based redaction by default through the `--set default-behavior` option, but only for a specific set of attributes, using the `--set included-sensitive-attribute` option:

   ```
   dsconfig set-log-field-syntax-prop \
        --syntax-name "Distinguished Name" \
        --set default-behavior:redact-value-components \
        --set included-sensitive-attribute:uid \
        --set included-sensitive-attribute:givenName \
        --set included-sensitive-attribute:sn \
        --set included-sensitive-attribute:cn \
        --set included-sensitive-attribute:mail
   ```

   > **Note**
   >
   > Unless overridden by a more specific log field behavior configuration, any `uid`, `givenName`, `sn`, `cn`, or `mail` attribute values that appear in DNs are redacted, while keeping the rest of the DN intact.
   > For example, a DN of `uid=jdoe,ou=People,dc=example,dc=com`, might be logged as `uid={REDACTED},ou=People,dc=example,dc=com`.

   For more information on including or excluding specific attributes and fields, see Log sanitization options.

2. To finalize your changes, restart the server:

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

Customizing log field behaviors

Use customized log field behaviors on a per-field basis to balance your organization's needs for logs that are both useful and secure.

About this task

You can use the following properties to configure log field behaviors for either predefined fields in either the `WriterBasedAccessLogFields` or `JSONAccessLogFields` or for custom log fields in third party plugins.

Behavior	Predefined field configuration property	Custom field configuration property
Preserve	`preserve-field`	`preserve-field-name`
Omit	`omit-field`	`omit-field-name`
Redact entire value	`redact-entire-value-field`	`redact-entire-value-field-name`
Redact value components	`redact-value-components-field`	`redact-value-components-field-name`
Tokenize entire value	`tokenize-entire-value-field`	`tokenize-entire-value-field-name`
Tokenize value components	`tokenize-value-components-field`	`tokenize-value-components-field-name`

For more information on log field behaviors, see Log sanitization options.

Steps

1. To create customized log field behaviors on a per-field basis, run `dsconfig create-log-field-behavior`.

   Example:

   The following example defines a log field behavior object so that the logger tokenizes the values of log fields that are expected to contain host names or IP addresses:

   ```
   dsconfig create-log-field-behavior \
       --behavior-name "Tokenize Hostnames" \
       --type text-access \
       --set tokenize-value-components-field:connect-from-address \
       --set tokenize-value-components-field:connect-to-address \
       --set tokenize-value-components-field:entry-rebalancing-source-server \
       --set tokenize-value-components-field:entry-rebalancing-target-server \
       --set tokenize-value-components-field:externally-processed-bind-end-client-ip-address \
       --set tokenize-value-components-field:requester-ip-address \
       --set tokenize-value-components-field:servers-accessed \
       --set tokenize-value-components-field:target-host
   ```

   Example:

The following example defines a log field behavior object so that the logger tokenizes the entire field value of a third party access log field named `myCustomField`:

```
dsconfig create-log-field-behavior \
    --behavior-name "My Log Field Behavior" \
    --type text-access \
    --set tokenize-entire-value-field-name:myCustomField
```

> **Note**
>
> Because the log field behavior does not define a default behavior, it only affects the manner in which specific fields are logged. The default behavior that is configured for the associated log field syntax is used for all other fields.

2. To associate the log field behavior you created with the loggers in which it should be used, run `dsconfig set-log-publisher-prop`.

   Example:

   The following example sets the default log field behavior created in the first example, `"Tokenize Hostnames"` for the file-based access logger:

   ```
   dsconfig set-log-publisher-prop \
       --publisher-name "File-Based Access Logger" \
       --set log-field-behavior:"Tokenize Hostnames"
   ```

3. To finalize your changes, restart the server:

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

Creating new log publishers

The 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.

   Example:

   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 `--set compression-mechanism: gzip` option to the previous command.
Because compression cannot be disabled or turned off after configured for the logger, plan carefully to determine your logging requirements, including log rotation and retention with regards to compressed logs.

2. View log publishers with the following command.

```
$ bin/dsconfig list-log-publishers
```

Creating a log publisher using dsconfig interactive command-line mode

Steps

1. In the command line, enter `bin/dsconfig`.
2. Authenticate to the server by following the prompts.
3. In the main menu, select the option to configure the log publisher.
4. In the `Log Publisher menu`, select the option to create a new log publisher.
5. Select the Log Publisher type.

   For this example, select File-Based Access Log Publisher.
6. Enter a name for the log publisher.
7. Enable it.
8. Enter the path to the log file relative to the PingDirectory server root.

   For this 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. In the `Log Publisher Properties menu`, select the option for `log-connects:false`, `log-disconnects:true`, `log-requests:false`, and `log-results:false`.
12. Enter `f` to apply the changes.

Configuring log rotation

The PingDirectory server allows you to configure the log rotation policy for the server.

About this task

When any rotation limit is reached, the 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.

  Example:

```
$ 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 server provides two log file rotation listener, the copy log file rotation listener and the summarize log file rotation listener, which you can enable with a log publisher.

About this task

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 is subject to deletion by a log file retention policy, but the copy is not automatically removed.

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.

Steps

- 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
```

> **Note**
>
> 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 are 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.

- 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
```

> **Note**
>
> 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

The server allows you to configure the log retention policy for each log on the server.

About this task

When a retention limit is reached, the server removes the oldest archived log before 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 server to retain. The default file count is 10 logs. If the file count is set to 1, then the log continues 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 server's requirements. This requires 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.

  Example:

```
$ bin/dsconfig set-log-publisher-prop \
  --publisher-name "File-Based Access Logger" \
  --set "retention-policy:Free Disk Space Retention Policy"
```

Configuring filtered logging

The PingDirectory server provides a mechanism to filter access log messages based on specific criteria.

About this task

You can use the filtered log with a custom log publisher to create and generate your own custom logs. Adding new filtered logs and associate publishers doesn't change the behavior of any existing logs. For example, adding a new log that only contains operations that were unsuccessful doesn't 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 didn't complete successfully. It then explains how to create a custom access log publisher that logs only operations matching that criteria.

> **Note**
>
> 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, such as result code, processing time, and response controls, turn off request logging and set the `include-request-details-in-result-messages` property to `TRUE`.

Because filtering based on the results of an operation can't 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. 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 didn't complete successfully.

   Example:

   ```
   $ 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.

   > **Note**
   >
   > The log rotation and retention policies are also set with this command.

   Example:

   ```
   $ 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 and verify that only information about failed operations was 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 you to 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 several properties, including:

- Protocol
- Client address
- Connection security
- Authentication state

Request Criteria

Defines sets of criteria for grouping and describing operation requests based on several properties, including:

- Properties for the associated client connection
- The type of operation
- Targeted entry
- Request controls
- Target attributes
- Other operation-specific terms

Result Criteria

Defines sets of criteria for grouping and describing operation results based on several properties, including:

- The associated client connection and operation request
- The result code
- Response controls
- Privileges missing or used

- Other operation-specific terms

Search Entry Criteria

Defines sets of criteria for grouping and describing search result entries based on several properties, including:

- The associated client connection and operation request
- The entry location and contents
- Included controls

Search Reference Criteria

Defines sets of criteria for grouping and describing search result references based on several properties, including:

- The associated client connection and operation request
- Reference contents
- Included controls

Configuring an Admin Alert Access Log publisher

Before configuring an Admin Alert Access Log, you must establish an administrative alert handler in your system.

Steps

1. To create a criteria object for the Admin Alert Access Log, use `dsconfig`.

   If you are using the `dsconfig` tool in interactive mode, the menu items for the criteria operations are located in the `Standard objects menu`.

   Example:

   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)`.

   ```
   $ 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. To create a log publisher of type `admin-alert-access`, use `dsconfig`.

   Example:

```
$ 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 PingDirectory server supports access logging using the syslog protocol that is part of the Berkeley Software Distribution (BSD) operating systems.

Syslog provides a flexible, simple means to generate, store, and transfer log messages that's supported on most UNIX and Linux operating systems.

The quasi-standard syslog message format cannot exceed 1 KB 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 and mail system messages. 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>`, and `<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.

You 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 so that messages with high severities are sent to a centralized repository while lower severity messages are stored locally on a server.

> **Note**
>
> Because 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 meaning of the PRI field is ambiguous.

Before you begin

You must identify the host name and port to which you want to connect.

Because the Syslog Protocol uses user datagram protocol (UDP) packets, you should use `localhost` and use some additional logging tools, such as `syslog-ng`. UDP is an 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 supported.

UDP-based communication is in the clear, so a network observer can see all of the log messages. You should only use syslog to log to a local syslog server and have it forward messages to a remote server in a secure manner. TLS encryption for TCP-based communication is optionally supported, so you can safely configure the server to log directly to a remote syslog server.

UDP does not provide any feedback about whether 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 are `Informational: informational` messages.

> **Note**
>
> This value is not standard across different types of UNIX or Linux systems. For more information, consult your 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 want to use. The default value for the `syslog-facility` property is `1` for user-level messages.

> **Note**
>
> These values are not standard across different types of UNIX or Linux systems. Consult your particular operating system documentation for properties specific to that system.

Facility	Description
`0`	kernel messages
`1`	user-level messages (default)
`2`	mail system

Facility	Description
`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 `queue-size` property determines the maximum number of log records that can be stored in the asynchronous queue.

The default queue size set is 10000, which means that the server continuously flushes messages from the queue to the log. The server does not wait for the queue to fill up before flushing to the log. Lowering this value impacts 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.

  Example:

```
$ 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 PingDirectory server provides an audit log, a specialized version of the access log, for troubleshooting problems that might occur during 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. Because the audit log adversely impacts the server's write performance, it is disabled by default.

Audit log format

The audit log uses standard LDIF format so that administrators can analyze what changes occurred to the data. The audit log begins logging when enabled and should be used to debug any issues that might have occurred.

Common properties include:

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 distinguished name (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

Enable the File-Based Audit Log Publisher using the `dsconfig` tool. The audit log impacts the PingDirectory server's write performance. Enable it only when troubleshooting.

Steps

- To enable the File-Based Audit Log Publisher, run `dsconfig`.

  Example:

  In this example, the instance name and startup ID are 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

About this task

Each value of an obscured attribute is replaced in the audit log with `***** OBSCURED VALUE *****`.

Steps

- To obscure the values of specific attributes in the audit log, use the `obscure-attribute` property.

  By default, attributes are not obscured because the values of password attributes appear in hashed form rather than in cleartext.

Managing the JDBC Access Log Publishers

You can configure the PingDirectory server to implement a centralized logging system with different databases by connecting to a database.

The PingDirectory server supports the Java Database Connectivity (JDBC) 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.

Centralized logging simplifies log correlation and analysis tasks and provides security by storing data in a single repository. However, data flow asymmetries might complicate synchronization or network provisioning and could unduly burden the central repository with heavy loads.

Before you begin

Before configuring the Java Database Connectivity (JDBC) Access Log Publisher, you must carry out two essential steps to set up the database.

To set up the database:

- Install the database drivers in the PingDirectory server `lib` directory.
- Define the log mapping tables needed to map access log elements to the database column data.

> **Note**
>
> Only those elements in the log mapping table get logged by the JDBC log publisher.

Configuring the JDBC drivers

About this task

The PingDirectory server supports several JDBC drivers available in the market. You should use the JDBC 4 drivers supported in the Java platform. For example, for Oracle databases, you must use the `ojdbc.jar` driver for Java and any associated JAR files (National Language Support `.jar`s 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 or files for your particular database and copy them into the `<server-root>/lib` directory.

Configuring the log field mapping tables

About this task

The log field mapping table associates access log fields with the database column names. Configure the log field mapping table using the `dsconfig` tool, which 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, you should map the following fields:

- `timestamp`
- `startupid`
- `message-type`
- `connection-id`
- `operation-type`
- `instance-name`

> **Note**
>
> The table name is not part of this mapping.

The PingDirectory server also provides these options that you can 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.
2. In the main menu, enter `o` to change to the `Standard Object menu`, and enter the number corresponding to `Log Field Mapping`.
3. In the `Log Field Mapping management menu`, enter the option to create a new Log Field Mapping.
4. In 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.
5. Enter a name for the new field mapping.

   For this example, enter `my-jdbc-test`.
6. In the `Access Log Field Mapping Properties menu`, select a property for which you want to change the value.

   Any property that is undefined is not logged by the JDBC Access Log Publisher.
7. Enter `f` to save and apply the changes.
8. In the `Log Field Mapping Management menu`, enter `q` to exit the menu.
9. View the existing Log Mappings on the system.

   Example:

   ```
   $ bin/dsconfig list-log-field-mappings
   ```

   Result:

   ```
   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` tool to configure the JDBC Access Log Publisher on the PingDirectory 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 PingDirectory server.
2. Launch the `dsconfig` tool in interactive command-line mode.

   Example:

   ```
   $ bin/dsconfig
   ```

3. Enter the connection parameters to bind to the PingDirectory server.

   Enter the host name or IP address, type of LDAP connection (LDAP, SSL, or StartTLS) that you are using on the PingDirectory server, the LDAP listener port number, the user bind DN, and the bind DN password.
4. In the main menu, enter the number corresponding to `Log Publisher`.
5. In the `Log Publisher management menu`, enter the option to create a new log publisher.
6. In the `Log Publisher template menu`, enter `n` to create a new Log Publisher.
7. In the `Log Publisher Type menu`, enter the option to create a new JDBC-Based Access Log Publisher.
8. Enter a name for the JDBC Access Log Publisher.
9. In the `Enabled Property menu`, enter the option to enable the log publisher.
10. In the `Server Property menu`, enter the option to create a new JDBC External Server.
11. Enter the name for the JDBC External Server.

    This is a symbolic name used to represent the database management system (DBMS).
12. In the `JDBC Driver Type Property menu`, enter the number corresponding to the type of JDBC database driver type.
13. Enter 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. Enter the host name or IP address (server-host-name) of the external server.
15. Enter the server listener port.

    For this example, enter `1541`.
16. Review the properties for the external server, and then enter `f` to apply the changes.
17. If you need to supply your own JDBC URL, enter `a` for advanced properties to open the `jdbc-driver-url` property and supply the appropriate URL.

    Example:

    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

PropertyValue(s)
---------------------------------------------------
1) description  -
2) jdbc-driver-type oraclethin
3) jdbc-driver-url jdbc:oracle:thin@myhost:1541:my_SID
4) database-namejdbc-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
```

Result:

When the JDBC Log Publisher is created, the PingDirectory server automatically generates a DDL file of the Log Field Mappings in the `<server-root>/logs/ddls/<name-of-logger>.sql` file, and you receive the following message.

```
JDBC External Server was created successfully
```

18. 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.

   Example:

   ```
   $ 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.

   Example:

```
$ 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"
```

Result:

When the JDBC Log Publisher is created, the PingDirectory server automatically generates a DDL file of the Log Field Mappings in the `<server-root>/logs/ddls/<name-of-logger>.sql` file.

3. 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 PingDirectory 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.

Modifying the File-Based Error Logs

Steps

- To modify the default File-Based Error Log, use `dsconfig`.

  Example:

```
$ 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 PingDirectory server supports a Syslog-Based Error Log Publisher using the same mechanism as the Syslog-Based Access Log Publisher.

Configure the error logger using the `dsconfig` tool.

Syslog error mapping

The PingDirectory server automatically maps error log severities to the syslog severities. The following mappings are used.

Error Log Severities	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

Configure a Syslog-based Error Log Publisher using the `dsconfig` tool. You should use syslog locally on `localhost` and use `syslog-ng` to send the data packets over the UDP protocol.

Because syslog implementations differ by vendor, review your particular vendor's syslog configuration.

Steps

- Use `dsconfig` to create a log publisher of type `syslog-based-error`.

  Example:

  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 PingDirectory server provides a File-Based Debug Log Publisher that can be configured when troubleshooting a problem that could occur during server processing.

Because the debug data might 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.

Use the `dsconfig` tool to create a debug log publisher.

> **Note**
>
> You should only create a debug logger when troubleshooting a problem because of the voluminous output the PingDirectory server generates.

Creating a File-Based Debug Log Publisher

Steps

- To create the debug log publisher, use `dsconfig`.

  The `log-file` property (required) sets the debug log path. You must also specify the rotation and retention policy for the debug log.

  Example:

```
$ 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

- To create the debug log publisher, use `dsconfig`.

Example:

```
$ bin/dsconfig delete-log-publisher \
  --publisher-name "File-Based Debug Logger"
```

Monitoring PingDirectory metrics with Splunk

Metrics from the PingDirectory 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.

A free Splunk (https://splunkbase.splunk.com/app/5523/) application is provided for the PingDirectory server that customers can use to create dashboards.

Sending PingDirectory metrics with StatsD

You can monitor the PingDirectory server using StatsD and Splunk.

In order to send metrics from PingDirectory 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 monitoring endpoint.

Steps

1. To enable the Stats Collector plugin, run the following command.

```
dsconfig set-plugin-prop \
    --plugin-name "Stats Collector" \
    --set enabled:true
```

2. To create a StatsD monitoring endpoint, run the following command. After running the command, the PingDirectory 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. Sign on 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. Click 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 PingDirectory server events to Splunk.

The Periodic Stats Logger plugin writes PingDirectory 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

The PingDirectory server can log various metrics, including LDAP operation counts and GC activity, to a JSON-formatted log.

About this task

To configure the PingDirectory 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 PingDirectory server app for Splunk

You can access several dashboards for monitoring the PingDirectory server.

When you have installed the PingDirectory 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.

Monitoring server metrics with Prometheus

Prometheus is an open-source monitoring application that can be used to track numeric metrics over time. It uses HTTP GET requests to retrieve the metric data in the OpenMetrics text-based format.

The server includes an HTTP servlet extension that can publish the values of a configured set of numeric monitor attributes in this format so that they can be consumed by a Prometheus server.

Enabling Prometheus support in the server

The server is preconfigured with an HTTP servlet extension that can publish a wide variety of metrics for use by Prometheus.

To enable the servlet extension, add it to an HTTP or HTTPS connection handler, and either disable and re-enable the connection handler or restart the server. For example:

```
dsconfig set-connection-handler-prop \
     --handler-name "HTTPS Connection Handler" \
     --add "http-servlet-extension:Prometheus Monitoring" \
     --set enabled:false

dsconfig set-connection-handler-prop \
     --handler-name "HTTPS Connection Handler" \
     --set enabled:true
```

The servlet extension offers several configuration properties that can be used to customize its behavior. They include:

`base-context-path`

The path that the Prometheus server can use to consume any published metrics. By default, this is `/metrics`.

`include-instance-name-label`

Indicates whether all published metrics should include an `instance` label whose value is the name of the server instance.

include-product-name-label

Indicates whether all published metrics should include a `product` label whose value is the name of the server product, such as `Ping Identity Directory Server` for the PingDirectory server.

include-location-name-label

Indicates whether all published metrics should include a `location` label whose value is the name of the configured location for the server instance.

always-include-monitor-entry-name-label

Indicates whether all published metrics should include a `monitor_entry` label whose value is the name of the monitor entry (the value of the `cn` attribute) from which the metric was taken.

> **Note**
>
> Even if this property is set to false, the server will still add this label to metrics in cases where there are multiple monitor entries with the same object class.

include-monitor-object-class-name-label

Indicates whether all published metrics should include a `monitor_object_class` label whose value is the name of the object class for the monitor entry from which the metric was taken.

include-monitor-attribute-name-label

Indicates whether all published metrics should include a `monitor_attribute` label whose value is the name of the monitor attribute from which the metric was taken.

label-name-value-pair

An optional set of name-value pairs for labels that should be included in all metrics published by the server. If provided, each item should use an equal sign to separate the label name from the value, such as `label_name=label_value`.

> **Important**
>
> Label names must:
>
> - Start with a letter or underscore
> - Only contain letters, digits, and underscores
> - Be unique
>
> Label values can be any non-empty string.

After the servlet extension has been enabled, you can verify that the metric information is available by retrieving it with a simple HTTP GET request using a web browser or a command-line tool like `curl` or `wget`.

For example, to retrieve the Prometheus metrics from the server `ds.example.com` using an HTTPS connection handler listening on port 8443 and the `default base-context-path` value of `/metrics`, use a URL of `https://ds.example.com:8443/metrics`.

Customizing published metrics

The server is preconfigured with a wide variety of definitions that publish the values of specified monitor attributes as metrics that can be consumed by Prometheus. However, you can also configure additional metrics by creating Prometheus monitor attribute metrics.

Each of these definitions includes the following configuration properties:

`metric-name`

The name of the metric as it should be published for use in Prometheus. The metric name must start with a letter or an underscore and must contain only letters, digits, and underscores. This is required, and all metric definitions must have a unique name.

`monitor-attribute-name`

The name of the monitor attribute whose value should be published as a Prometheus metric. This is required, and only single-valued attributes are supported. The value must be an integer or floating-point number.

`monitor-object-class-name`

The name of the object class for the monitor entry that contains the attribute to publish as a Prometheus metric. This is required.

`metric-type`

The data type for the metric to be published. This is required, and the value should be one of the following:

- `boolean` — Indicates that the value of the attribute represented by `monitor-attribute-name` has one of the following values: `true`, `false`, `enabled`, `disabled`, `yes`, `no`, `on`, `off`, `1`, or `0`. The server sends a gauge metric to Prometheus with a value of:
  - `1` to represent any of the following: `true`, `enabled`, `yes`, `on`, or `1`
  - `0` to represent any of the following: `false`, `disabled`, `no`, `off`, or `0`
- `counter` — Indicates that the value is one that increases over time as certain events occur in the server. This can include values that simply increment each time the event occurs, such as counting the number of connections accepted by the server since it started, but it can also include values that increase by larger amounts, such as counting the total number of bytes transferred in responses.
- `gauge` — Indicates that the value is one that can go up or down over time and whose value represents the current state of some aspect of the server, such as the number of connections that are currently established.

`filter`

An optional search filter that can be used to indicate that the metric should only be published for monitor entries that match this filter. If this is not specified, then the metric is published for any monitor entry with the specified attribute and object class.

`metric-description`

An optional human-readable string that describes the purpose for the metric or that provides some other additional information about it.

`label-name-value-pair`

An optional set of name-value pairs for labels that should be included in the definition for the metric. If provided, each item should use an equal sign to separate the label name from the value, such as `label_name=label_value`. Label names must start with a letter or underscore and must only contain letters, digits, and underscores. Label values can be any non-empty string.

> **Note**
>
> Metric-specific labels override server-wide labels, so if a metric-specific label is defined with the same name as a label that would otherwise be applied to all metrics, then the metric-specific value will be used instead of the server-wide value.

For example, to create a Prometheus metric that exposes the value of the `currentConnections` attribute of the general monitor entry, you could use a configuration change like:

```
dsconfig create-prometheus-monitor-attribute-metric \
        --extension-name "Prometheus Monitoring" \
        --metric-name general_monitor_current_connections \
        --set monitor-attribute-name:currentConnections \
        --set monitor-object-class-name:ds-general-monitor-entry \
        --set metric-type:gauge \
        --set "metric-description:The number of connections currently established"
```

If you want to remove any of the metrics that are included in the out-of-the-box configuration, then remove the definition for that metric from the server configuration.

Consuming metrics with Prometheus

After the server has been configured to publish its metrics, you can configure Prometheus to consume them.

The basic process is to update the server's `prometheus.yml` file to add a `job` element to the `scrape_configs` section for the server from which the metrics should be retrieved, as in the following example:

```
- job_name: "ds.example.com:8443"
  metrics_path: "/metrics"
  scheme: "https"

  tls_config:
    ca_file: ds.example.com-ca-certificate.pem

  static_configs:
    - targets: ["ds.example.com:8443"]
```

See the Prometheus documentation for complete details.

Monitoring server metrics with ELK

You can use Elasticsearch, Logstash, and Kibana, referred to together as the ELK Stack, to monitor a PingDirectory server by consuming the server logs.

The ELK Stack is highly configurable and can support server monitoring needs in production environments. To work with PingDirectory servers, point Logstash to the location of the JSON-formatted server logs that you want to consume.

> **Note**
>
> The ELK Stack is a third-party solution provided by Elastic. The information in this topic is for your convenience and should be confirmed by consulting the manufacturer.

If you want to try monitoring PingDirectory using the ELK Stack with minimal configuration:

- Use a test environment.
- Create log publishers in JSON formats to make them easy to parse.
- Because Filebeat is easier to configure, use it to consume your logs instead of Logstash. Configure Filebeat with the location of the PingDirectory logs and a log output type of JSON.
- Consult the Elastic documentation for help configuring and deploying an ELK Stack in Docker.

If you have an established PingDirectory logging configuration and want to develop a production-capable ELK monitoring solution:

- Start in a lower environment.
- Because it is more configurable, try working with Logstash. For example, you can create filters on your logging data. If possible, use JSON-formatted logs.
- If you have a clustered server configuration, configure all PingDirectory logs to write to a central location from which Logstash can consume them.
- Determine what kinds of data you want to surface in your monitoring and create Elasticsearch indexes around them. For example, you might want to focus on particular types of searches or updates.

Managing notifications and alerts

The PingDirectory server provides delivery mechanisms for account status notifications and administrative alerts using SMTP, Java Management Extensions (JMX), or SNMP in addition to standard error logging.

> **Important**
>
> SNMP is deprecated.

Alerts and events reflect state changes within the server that might 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.

> **Note**
>
> Account status notifications are only delivered to the account owner, notifying a change in state in the account.

Account status notifications

The PingDirectory server supports notification handlers that can notify users, administrators, or both of significant changes related to password policy state for user entries.

The following notification handlers are available:

Error Log Account Status Notification Handler

Sends alerts to the error log when an account event occurs. Enabled by default.

SMTP Account Status Notification Handler

Sends notifications to designated email address. Disabled by default, but you can enable it using the `dsconfig` command.

Account status notification types

Notification handlers send alerts when one of the following account status events occurs during password policy processing.

Account Status Notification Types

Account Status Notification Types	Description
`account-disabled`	Generates a notification when a user account is disabled by an administrator.
`account-enabled`	Generates a notification when a user account is enabled by an administrator.
`account-expired`	Generates a notification when a user authentication attempt fails because the account has expired.
`account-idle-locked`	Generates a notification when a user authentication attempt fails because the account has been locked after idling for too long.
`account-permanently-locked`	Generates a notification when a user account is permanently locked, requiring administrative action to unlock the account after too many failed authentication attempts.
`account-reset-locked`	Generates a notification when an authentication attempt fails because the user account is locked after the user failed to change a password that was reset by an administrator within the required interval.
`account-temporarily-locked`	Generates a notification when a user account is temporarily locked after too many failed attempts.

Account Status Notification Types	Description
`account-unlocked`	Generates a notification when a user account is unlocked by an administrator.
`password-changed`	Generates a notification when a user changes his or her own password.
`password-expired`	Generates a notification when a user authentication fails because the password has expired.
`password-expiring`	Generates a notification the first time a password expiration warning is encountered for a user password.
`password-reset`	Generates a notification when 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.

You can view the log at `logs/error`.

Disabling the Error Log Account Status Notification Handler

Steps

- To disable the Error Log Handler, use the `dsconfig` tool.

```
$ bin/dsconfig set-account-status-notification-handler-prop \
  ---handler-name "Error Log Handler" --set enabled:false
```

Removing a notification type from the Error Log Handler

Although you can remove an account status notification type from the Error Log Handler, it's not recommended.

About this task

For a list of notification types, see Account status notification types.

Steps

- To remove an account status notification type, use the `dsconfig` tool with the `--remove` option.

  Example:

```
$ 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 PingDirectory 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.

Configuring the SMTP server

Steps

1. To configure a simple mail server, use the `dsconfig` command.

   Example:

   ```
   $ bin/dsconfig create-external-server --server-name smtp1 \
     --type smtp --set server-host-name:smtp.example.com
   ```

2. To configure an SMTP server, use the `dsconfig` command.

   Example:

   ```
   $ bin/dsconfig set-global-configuration-prop \
     --set smtp-server:smtp1
   ```

   This command adds the server to the global list of mail servers that the PingDirectory server can use.

Configuring a StartTLS connection to the SMTP server

Steps

1. To configure a StartTLS connection to the server, use the `dsconfig` command.

   Example:

   ```
   $ 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. To configure a newly-created SMTP server, use the `dsconfig` command.

   Example:

   ```
   $ bin/dsconfig set-global-configuration-prop \
     -set smtp-server:myTLSServer
   ```

   This command adds the server to the global list of mail servers that the PingDirectory server can use.

Configuring an SSL connection to the SMTP server

Steps

1. To create an external SMTP server using SSL, run the `dsconfig` command.

   Example:

   ```
   $ 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. To configure an SMTP server, run the `dsconfig` command.

   Example:

   ```
   $ bin/dsconfig set-global-configuration-prop \
     --set smtp-server:ssl.smtp.example.com
   ```

   This command adds the server to the global list of mail servers that the PingDirectory server can use.

Enabling the SMTP account status notification handler

Steps

- To enable the SMTP account status notification handler, use the `dsconfig` command.

  Example:

```
$ 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
```

Viewing the account status notification handlers

After you enable the SMTP server, you can view the list of account status notification handlers.

Steps

- To view the list of account status notification handlers, run the `dsconfig` command.

  Example:

```
$ bin/dsconfig list-account-status-notification-handlers
```

  Result:

```
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 when password policy state changes occur in the server, you must configure the password policy that governs the entry being updated to use one or more account status notification handlers.

About this task

By default, password policies are not configured with such handlers, so no account status notifications are generated.

The `account-status-notification-handler` property controls the set of account status notification handlers for a password policy. You can be configure that property using `dsconfig` or the admin console.

Steps

- To configure the `account-status-notification-handler` notification handler property, run `dsconfig`.

  Example:

```
$ bin/dsconfig set-password-policy-prop \
  --policy-name "Default Password Policy" \
  --set "account-status-notification-handler:Error Log Handler"
```

This command updates the default password policy to trigger the error log account status notification handler for any appropriate password policy state changes.

Administrative alert handlers

The 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 PingDirectory provides several 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 PingDirectory. 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 primary 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

The PingDirectory server can generate administrative alerts when the events occur.

A full listing of system alerts and their severity is available at `<server-root>/docs/admin-alerts-list.csv`.

Changing the timeout for an exec alert handler

Exec alert handlers automatically terminate their associated commands after the default duration. You can change this duration or disable the timeout if needed.

About this task

The `command-timeout` property for an exec alert handler has a default value of 1 hour, after which the handler terminates any commands or scripts it initiated and logs an error. This timeout helps mitigate the effects of a command or script running longer than expected.

Depending on your configuration for a given exec alert handler, you might want to adjust this value up or down. You can disable the timeout by setting the property's value to `0 s`.

Steps

- Set the value for the `command-timeout` property of an exec alert handler.

  Choose from:

  - Create an exec alert handler with the desired timeout value. For example:

    ```
    $ bin/dsconfig create-alert-handler \
      --handler-name TestAlertHandler \
      --type exec \
      --set enabled:true \
      --set command:testScript.sh \
      --set "command-timeout:2 h"
    ```

  - Modify the `command-timeout` property of an existing exec alert handler. For example:

    ```
    $ bin/dsconfig set-alert-handler-prop \
      --handler-name TestAlertHandler \
      --set "command-timeout:15 m"
    ```

  - Disable the `command-timeout` property of an existing or new exec alert handler by setting the value to `0`. This allows any initiated commands or scripts to run indefinitely. For example:

    ```
    $ bin/dsconfig set-alert-handler-prop \
      --handler-name TestAlertHandler \
      --set "command-timeout:0 s"
    ```

Configuring the JMX connection handler and alert handler

You can configure the Java Management Extensions (JMX) connection handler and the alert handler using the `dsconfig` tool.

To allow users to receive JMX notifications, enable the `jmx-read` and `jmx-notify` privileges. By default, these privileges aren't granted to any users, including root users or global administrators.

For security reasons, you should create a separate user account that has only these privileges.

You can also configure the JMX connection handler and the alert handler using `dsconfig` in interactive command-line mode, which is visible in the `Standard object menu`.

Configuring the JMX connection handler

About this task

To enable the JMX connection handler:

Steps

1. Run `dsconfig`.

   Example:

   ```
   $ 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 using the `ldapmodify` tool using an LDIF representation.

   Example:

   ```
   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

About this task

To configure the Java Management Extensions (JMX) alert handler:

Steps

- Run `dsconfig`.

  Example:

```
$ bin/dsconfig set-alert-handler-prop --handler-name "JMX Alert Handler" \
  --set enabled:true
```

Configuring the SMTP alert handler

To create a new instance of an SMTP alert handler, use the `dsconfig` tool.

About this task

By default, there is no configuration entry for an SMTP alert handler.

Steps

- Use the `dsconfig` tool to configure the SMTP Alert Handler.

  Example:

```
$ 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 in the `Standard object menu`.

About this task

Before you begin, you need an SNMP subagent capable of communicating using SNMP2c. For more information about SNMP, see Monitoring using SNMP.

The PingDirectory server also supports an SNMP alert handler, which is used in deployments that do not enable an SNMP subagent.

To configure the SNMP subagent alert handler:

Steps

- Run `dsconfig`.

  Example:

  In this example:

  - 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.

```
$ 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
```

Email account status notification handler

The e-mail account status notification handler enables the server to send messages that can contain either a plain-text body, an HTML-formatted body, or both.

If you configure the server to send both plain-text and HTML-formatted bodies, 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 the server can use to generate multi-part email messages to send to end users and server administrators when certain events occur.

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 because of 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 because of 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

An email message for the email account status notification handler contains the following structure:

- 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 can 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 can 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.

Header section

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 can contain several 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.

> **Tip**
>
> If you don't want a message sent for a particular notification type, don't 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 is visible to all recipients.

The value can be either an email address, such as `jdoe@example.com`, or a name followed by the email address in angle brackets, such as `John Doe <jdoe@example.com>`.

To specify multiple `TO` recipients, specify this element 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 isvisible to all recipients.

The value can be either an email address, such as `jdoe@example.com`, or a name followed by the email address in angle brackets, such as `John Doe <jdoe@example.com>`.

To specify multiple `CC` recipients, specify this element multiple times.

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 is not visible to other recipients.

The value can be either an email address, such as `jdoe@example.com`, or a name followed by the email address in angle brackets, such as `John Doe <jdoe@example.com>`.

To specify multiple `BCC` recipients, specify this element multiple times.

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, such as `jdoe@example.com`, or a name followed by the email address in angle brackets, such as `John Doe <jdoe@example.com>`.

This element must be specified exactly once.

REPLY-TO

Specifies the default address to which users should send replies.

The value can be either an email address, such as `jdoe@example.com`, or a name followed by the email address in angle brackets, such as `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 for 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, such as `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.

Body sections

The header section must be followed by plain-text body sections, HTML-formatted body sections, or both. 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, you should add a plain-text body 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 are removed, but blank lines in the middle of the body are preserved.

Attachment sections

The template can contain any number of optional attachment sections that describe any attachments that should be included in the message. Specify each attachment 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 present, this value must be either `true` or `false`. If it is omitted, a default value of `false` is assumed.

If `true`, indicates that the attachment should be an inline attachment. For example, if you include an image that should be referenced within the HTML body, the image source path should be `cid:` followed by the filename, such as `cid:company-logo.png`.

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. This enables you to customize the message content with information from the account status notification, details from the associated user entry, and other information. Learn more about Apache Velocity.

The following variables are defined for use with custom directives.

Variable	Description
`$account_entry`	The entry for the user associated with the account status notification.
`$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.

Variable	Description
`$before_entry`	**A version of the associated entry as it appeared before the operation was processed.** **Note** This is only available for notifications generated as a result of modify or modify distinguished name (DN) operations.
`$after_entry`	**A version of the associated entry as it appeared after the operation was processed.** **Note** 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.

Directive	Syntax	Description
`#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.**

Directive	Syntax	Description
`#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.

Directive	Syntax	Description
`#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.

Directive	Syntax	Description
`#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.

Directive	Syntax	Description
`#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.

Directive	Syntax	Description
`#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. See the following table for ***propertyName*** values.

Directive	Syntax	Description
`#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. See the following table for *propertyName* values.
`#getFormattedTimestamp`	*timestamp*, *formatString*, *variableName*	Retrieves a representation of the provided timestamp, which must be in generalized time form, formatted with the provided 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 provided 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.

Property	Description
`notification-type`	The name of the account status notification type.

Property	Description
`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`	The value is `true` if the account is considered usable, or `false` if it is not usable. The account is considered unusable if, for example, the account is locked, expired, or not yet active; if the password is expired; if the user must choose a new password; and so on.
`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 is `true` if the account has been administratively disabled, or `false` if it has not been administratively disabled.
`account-is-not-yet-active`	The value is `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.

Property	Description
`seconds-until-account-activation`	The number of seconds until the account becomes active if it has an activation time 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 in the future.
`seconds-since-account-activation`	The number of seconds since the account became active if it has an activation time 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 in the past.
`account-is-expired`	The value is `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 is `true` if the account has been temporarily or permanently locked because of 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 because of too many failed authentication attempts.
`seconds-until-unlock`	The number of seconds until the account unlock time, if applicable.

Property	Description
`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 is `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 can pass between successful authentications before an account becomes idle-locked, if configured.
`idle-lockout-interval`	A human-readable duration that indicates how much time can 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 is `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.

Property	Description
`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 is `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 is `true` if the account had been locked but was just unlocked by the associated operation, or `false` if not.
`password-is-expired`	The value is `true` if the account has an expired password, or `false` if not.
`password-is-expiring`	The value is `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 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 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 in the past.

Property	Description
`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 PingDirectory 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. 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 the server in the `config/alerts.ldif` file so they can survive server restarts. By default, the alerts remain in the system for seven days before being removed. Administrators can configure the number of days alerts are retained by 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 admin console.

Viewing information in the Alerts Backend

About this task

To view admin alerts:

Steps

- Run `ldapsearch`.

  Example:

  ```
  $ bin/ldapsearch --port 1389 --bindDN "cn=Directory Manager" \
    --bindPassword secret --baseDN cn=alerts "(objectclass=ds-admin-alert)"
  ```

  Result:

  ```
  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
  ```

Modifying the alert retention time

Steps

- To change the retention time of admin alerts, run `dsconfig`.

  Example:

  ```
  $ bin/dsconfig set-backend-prop --backend-name "alerts" \
    --set "alert-retention-time: 2 weeks"
  ```

  After this time, the information is purged from the PingDirectory server. The minimum retention time is 0 milliseconds, which immediately purges the alert information.

- To view the `alert-retention-time` property, use `dsconfig`.

  Example:

  ```
  $ bin/dsconfig get-backend-prop --backend-name "alerts" \
    --property alert-retention-time
  ```

  Result:

```
Property: Value(s)
---------------------:---------
alert-retention-time : 2 w
```

Configuring duplicate alert suppression

About this task

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 that can be sent over the administrative alert framework in the time limit specified in the `duplicate-alert-time-limit` property.

To configure the maximum number of times an alert is generated within a particular timeframe for the same condition:

Steps

- Run `dsconfig`.

  Example:

```
$ bin/dsconfig set-global-configuration-prop \
  --set duplicate-alert-limit:2 \
  --set "duplicate-alert-time-limit:3 minutes"
```

Working with alarms, alerts, and gauges

Alarms, alerts, and gauges alert administrators to changes in server conditions that might require attention.

Alarms

An alarm represents a stateful condition of the server or a resource that might indicate a problem, such as low disk space or external server unavailability.

Alarms have severity, name, and message. Alarms will always have a Condition property, and can have a Specific Problem or Resource property. If surfaced through SNMP, a Probable Cause property and Alarm Type property are also listed. You can configure alarms to generate alerts when the alarm's severity changes.

You can configure the Alarm Manager, which governs the actions performed when an alarm state is entered, through the `dsconfig` tool and the admin console. A complete list of system alerts, alarms, and their severity is available in `<server-root>/docs/admin-alerts-list.csv`.

The server complies with the International Telecommunication Union CCITT Recommendation X.733 (1992) standard for generating and clearing alarms. If configured, entering or exiting an alarm state might 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. You can view alarms with the `status` tool. As with other alert types, you can configure alert handlers can to manage the alerts generated by alarms.

Alerts

There are two alert types supported by the server: standard and alarm-specific.

The server constantly monitors for conditions that might need administrator attention, such as low disk space. For this condition, the standard alert is `low-disk-space-warning`, and the alarm-specific alert is `alarm-warning`.

You can configure the server to generate alarm-specific alerts as well as standard alerts. By default, standard alerts are generated for conditions internally monitored by the server. However, gauges can only generate alarm-alerts.

Gauges

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.

Numeric gauges monitor continuous values like CPU load or free disk space. Indicator gauges monitor enumerated set of values such as `server available` or `server unavailable`. Gauges generate alarms when the gauge's severity changes because of changes in the monitored value.

The server installs a set of gauges that are specific to the product and that can be cloned or configured through the `dsconfig` tool. You can tailor existing gauges to fit each environment by adjusting the update interval and threshold values. Configuration of system gauges determines the criteria by which alarms are triggered.

Use the Stats Logger to view historical information about the value and severity of all system gauges. For more information, see Profiling server performance using the Stats Logger.

Viewing information in the Alarms Backend

About this task

Use `ldapsearch` to view alarms.

Steps

- To display the listing for the CPU usage alarm, run the following command.

```
$ bin/ldapsearch --port 1389 --bindDN "cn=Directory Manager" \
  --bindPassword secret --baseDN cn=alarms "(objectclass=ds-admin-alarm)"
```

Result:

```
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, you can manually increase the severity of a gauge to verify that the server takes the appropriate action when an alarm state change.

You can then use the `status` tool to verify alarms and alerts.

Testing alarms and alerts

Steps

1. Use `dsconfig` to configure a gauge and set the `override-severity` property to critical.

   The following example configures the CPU Usage (Percent) gauge.

   Example:

   ```
   $ 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.

   Example:

   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

The server raises indeterminate alarms for a server condition for which a severity cannot be determined.

In most cases these alarms are benign and do not issue alerts, nor do they appear in the output of the `status` tool or admin 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 might produce indeterminate alarms if a 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 might 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. A search of the monitor backend for this gauge's entry results in an error message that might 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
              ...
```

Directory REST API configuration

The Directory REST API is the native interface for client access to the PingDirectory server.

Overview

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. Learn more in the PingDirectory REST API Reference and the Developer portal.

The Directory REST API gives developers who are more comfortable with REST than LDAP access to arbitrary directory data in a way that ensures that 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.

Although 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's a less common standard for identity stores, it might not offer as much functionality or be as user-friendly as the Directory REST API.

> **Tip**
>
> You can define ACIs in PingDirectory that map to specific OAuth scopes. Learn more in Using OAuth scopes for ACI rules with the REST API.

Configuration options

You can 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. To disable basic authentication, enter the following command:
	`dsconfig set-http-servlet-extension-props \`
	`  --extension-name "Directory REST API" \`
	`  --set basic-auth-enabled:false`

`identity-mapper`	If HTTP basic authentication is enabled, the identity mapper referenced by this distinguished name (DN) must be used to map the 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.

Command	Description
`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 because of 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 `/directory/v1/schemas` to retrieve the schemas of all available object classes in the REST API. By default, no schemas are returned. Changes to this value take effect immediately.

Example:

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
```

Using OAuth scopes for ACI rules with the REST API

To help isolate access to admin credentials in authentication workflows, you can use OAuth scopes to enforce ACIs for users authenticating to specific Directory REST API endpoints.

About this task

When users make requests to Directory REST API endpoints, they are required to authenticate. Depending on the configured authentication policy, they could be prompted for credentials and issued an OAuth 2.0 bearer token in exchange.

After receiving the token, users can be granted OAuth scopes by a token validator, such as PingFederate. PingDirectory can then apply user-configured ACIs to those scopes, enabling the REST API to provide the permissions for the user and the request.

Important

- You must have an encryption settings definition configured on the server before using OAuth scopes with the Directory REST API. Learn more in Creating encryption settings definitions.
- In a proxied configuration, PingDirectoryProxy can still pass OAuth scopes to PingDirectory for REST API requests.
- Some Directory REST API endpoint requests don't support this feature.

Creating ACIs with OAuth scopes

Executing a request to the REST API using an OAuth bearer token depends on both the configured ACIs in the PingDirectory server and the scopes used to authenticate the request present in the provided OAuth bearer token.

To enable PingDirectory to apply ACIs to specific OAuth scopes, you need to define these ACIs with the desired scopes using the `oauthscope` bind rule. Learn more about ACIs in ACI syntax and ACI bind rules.

Example

You can add the following ACI to PingDirectory to allow all authenticated users with the OAuth scope `example:scope` to perform the Get Password Quality Requirements extended operation:

```
(extop="1.3.6.1.4.1.30221.2.6.43")(version 3.0; acl "Extended op permissions for get password quality
requirements"; allow (read) oauthscope="example:scope";)
```

With the ACI added, the authenticating user can send a `POST` request to the `/directory/v1/passwordRequirements` endpoint, which performs this extended operation on the backend. You can find details about this endpoint in Get Password Quality Requirements.

If the `example:scope` OAuth scope is validated in the bearer token included in the HTTP Authorization Header of the request (and no other ACIs explicitly deny the user's privilege to perform the extended operation), then the user will get a successful HTTP response.

Managing Server SDK extensions

The 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 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 can 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 several different types of extensions for Ping Identity server products, including the PingDirectory server, the PingDirectoryProxy server, and the 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

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-primary 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, PingDirectory 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. Before 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.

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 PingDirectory servers as pets today can take advantage of current DevOps and IaC principles to turn them into cattle.

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 Amazon Web Services (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.

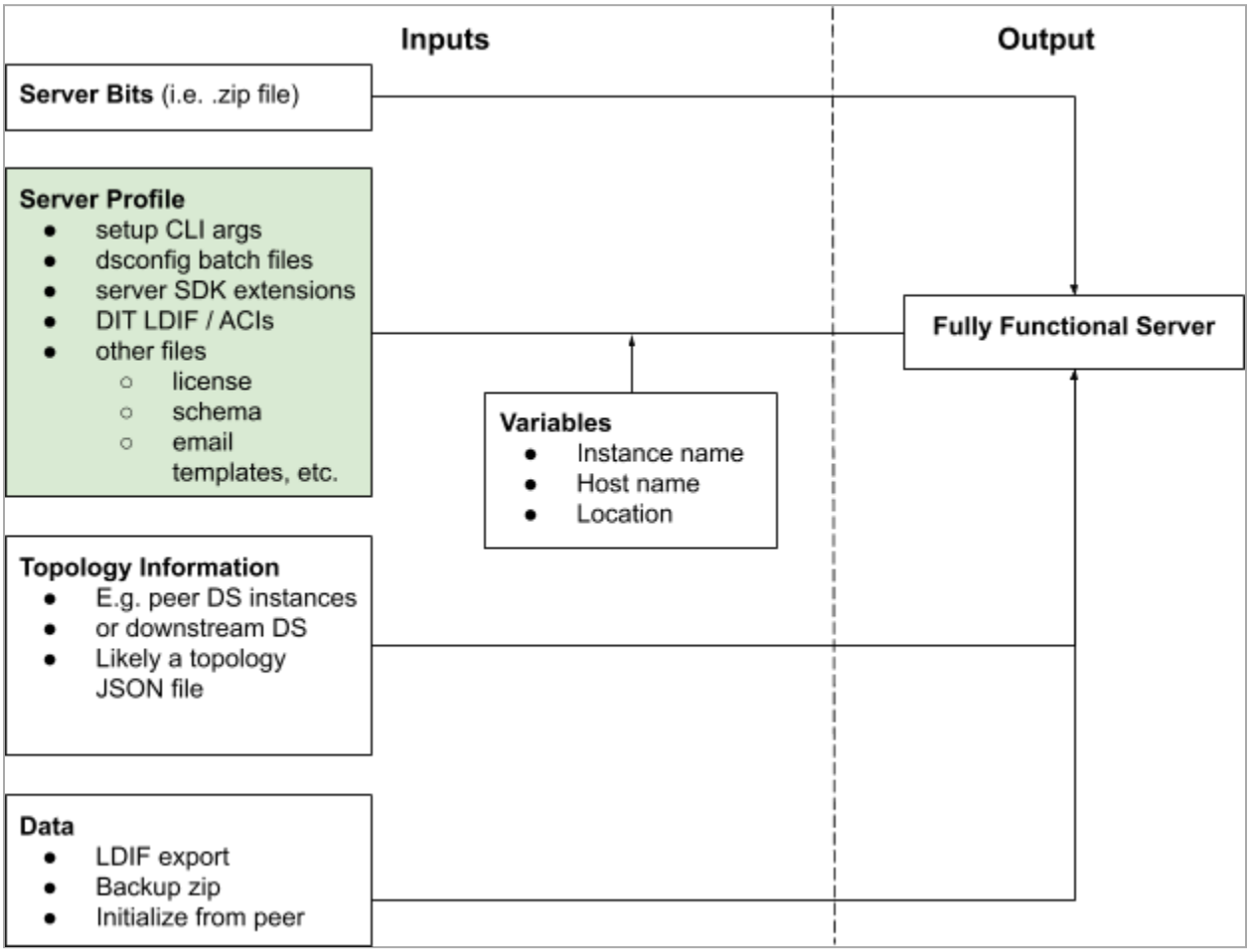
Server profiles

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.
- 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 before 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.

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

Built-in variable	Description
`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 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 `setup-arguments.txt` file, located in the `resources/server-profile-template.zip` archive.

When you run the **`manage-profile setup`** command, these arguments are passed to the server's setup tool. To view the arguments available in this file, run the server's **`setup --help`** command.

To provide the equivalent, non-interactive command-line interface (CLI) arguments after any prompts have been completed, run **`setup`** interactively. The `setup-arguments.txt` file in the server profile template contains an example set of arguments that can be changed.

`setup-arguments.txt` is the only required file in the profile.

dsconfig/ and pre-setup-dsconfig/

You can include **`dsconfig`** batch files in the server profile with information about changes to apply to the out-of-box system configuration. Any batch files in the server profile must include a `.dsconfig` extension. **`dsconfig`** batch files in the server profile can't make changes to any topology configuration, including topology admins and server instances, so you should make topology configuration changes outside of the server profile.

The batch files can be placed in the following server profile directories:

`dsconfig/`

Contains information about changes to apply after the **`manage-profile`** tool runs **`setup`**, copies any post-setup files into place, and installs any Server SDK extensions contained in the server profile. The `dsconfig/` directory should be used for most configuration changes applied to the server.

`pre-setup-dsconfig/`

Holds information about changes to apply immediately before the `manage-profile` tool runs `setup`. You should only use the `pre-setup-dsconfig/` directory when setting up the server with a pre-existing encryption settings database and applying the changes required for configuring and activating the cipher stream provider needed to access the encryption settings database.

> **Note**
>
> If multiple batch files are present in the directory that the **`manage-profile`** tool is using, the files are processed in ascending lexicographic order. For example, **`00-base.dsconfig`** is processed before **`01-second.dsconfig`**.

> **Tip**
>
> You can use the **`config-diff`** tool to create a **`dsconfig`** batch file that reproduces the current topology configuration.

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 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.

The `--excludeSetupArguments` option generates a server profile without a setup-arguments.txt file. This is useful when generating server profiles for use with Docker images.

manage-profile setup

To apply a server profile to an unconfigured server, use the `setup` subcommand, which replaces the normal setup tool when using a server profile. Run `manage-profile setup` to do the following:

1. Copy the `server-root/pre-setup` files to the server root.
2. Apply changes for any batch files contained in the `pre-setup-dsconfig` directory.
3. Run the setup tool.
4. Copy the `server-root/post-setup` files to the server root.
5. Install any Server SDK extensions.
6. Apply changes for any `dsconfig` batch files.
7. Import any LDIF files contained in the `ldif` directory structure.
8. Start the server.

`manage-profile setup` creates a copy of the profile in a temporary directory specified by the `--tempProfileDirectory` option. The tool leaves the server running upon completion unless you use the `--doNotStart` option.

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.

The `--skipValidation` option skips the validation step when running on an offline server

> **Important**
>
> When you run the `manage-profile replace-profile` tool with an SDK extension included in the new profile, the tool invokes the `setup` command.
> The `manage-profile replace-profile` tool can update the server version when needed, but will fail if you attempt to downgrade the server to an earlier version. It can also 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.

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 primary when you do not have a quorum majority of online severs.

  The topology registry uses a primary-secondary architecture in which all write requests are chained through a single write-primary. When a primary could not be nominated, the topology becomes read-only. To work around this issue, a single server had to be forced as a primary 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.

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.

   Example:

   ```
   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.

   Example:

   ```
   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.

   Example:

```
dsreplication enable \
  --topologyFilePath /path/to/topology.json \
  --retryTimeoutSeconds 120
```

5. To initialize the replication data, run `dsreplication` with the `initialize` option.

   Example:

```
dsreplication initialize \
  --topologyFilePath /path/to/topology.json
```

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 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. The contents of the `config/sample-dsconfig-batch-files/create-topology-admin-user.dsconfig` batch file are shown below:

```
# Although setup automatically creates an initial root user account that can
# be used to manage the server, we strongly recommend creating a separate
# account for each administrator rather than using a single shared account.
# This offers several benefits, including:
#
# * It is easier to determine which administrator performed a given action.
# * There is no need to share credentials or coordinate password changes.
# * It is easier to use strong authentication options like certificates or
#   two-factor mechanisms.
# * You can customize the level of access that each administrator has.
#
# We also recommend creating topology admin user accounts rather than root
# user accounts.  If you create a root user account, then it is only created
# in that one instance.  If you create a topology admin user, then that
# account will be available in all instances in the topology.
#
# Because administrative accounts are very powerful, they are high-priority
# targets for attackers to try to compromise.  They should be required to have
# strong credentials, and you may wish to require that they use strong
# authentication mechanisms (see other dsconfig batch files for configuring
# password validators and other password policy features).  You may also want
# to give them usernames that are not likely to be guessed by an attacker,
# even if they know information about the individuals administering the
# service.  For example, you may not want to use their name in the DN or
# username, choosing instead something that is less predictable or even
# completely random like a UUID.
#
# Once each administrator has their own account, we recommend that you disable
# or remove the initial root user created during setup.  See the
# disable-or-remove-the-initial-root-user.dsconfig batch file for more
# information about that.

#
# NOTE:  Do not edit this file directly.  Changing this file could prevent it
# from being updated during a server upgrade.  If you want to alter the
# dsconfig commands below before applying the configuration changes, copy this
# file to another directory and edit that copy.
#

# Create a new topology administrator account.  In this example, the user will
# be able to read the configuration but will not be allowed to update it, and
# will be given the bypass-read-acl privilege rather than bypass-acl, which
# only ensures they will be able to read entries but they will not be
# permitted to update them unless permitted by the server's access control
# configuration.  They will also only be permitted to communicate with the
# server over a secure connection, and they will only be permitted to
# authenticate with the EXTERNAL or UNBOUNDID-TOTP mechanisms.
dsconfig create-topology-admin-user \
     --user-name "[USER_NAME]" \
     --set "password:[PASSWORD]" \
     --set "first-name:[FIRST_NAME]" \
     --set "last-name:[LAST_NAME]" \
     --set "user-id:[USER_NAME]" \
     --set inherit-default-root-privileges:true \
```

```
    --set privilege:bypass-read-acl \
    --set privilege:-bypass-acl \
    --set privilege:-config-write \
    --set search-result-entry-limit:0 \
    --set time-limit-seconds:0 \
    --set look-through-entry-limit:0 \
    --set idle-time-limit-seconds:0 \
    --set require-secure-authentication:true \
    --set require-secure-connections:true \
    --set "allowed-authentication-type:SASL EXTERNAL" \
    --set "allowed-authentication-type:SASL UNBOUNDID-TOTP"
```

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.

Scaling down

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 PingDirectory 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. Extract 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
   ```

Command-line tools

The server provides a full suite of command-line tools necessary for administration. 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

PingDirectory provides the following command-line tools, which you can run in interactive, non-interactive, 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`

Command-Line Tools

`audit-data-security`	Invoke data security audit processing in order to identify potential risks or other notable security characteristics contained in directory data. This tool schedules an internal task with the server 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. The list of available auditors can be obtained using `dsconfig list-data-security-auditors --advanced --property name`. Either the `--includeAuditor` or the `--excludeAuditor` arguments can be used to limit the scope of the audit. Additionally, the entries scanned can be limited by specifying the backends to scan, or by specifying an LDAP filter that is used to selected entries to be processed. This tool schedules an operation to run within the PingDirectory server's process. LDAP connection options must be supplied 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.
`authrate`	Perform repeated authentications against the PingDirectory server, where each authentication consists of a search to find a user followed by a bind to verify the credentials for that user.

`backup`	Back up one or more directory server backends. Each backend backup is stored in a separate backend backup directory. A backend backup directory can contain multiple backups of the backend. Each backend backup directory contains a `backup.info` file providing information about each backup in the directory and an archive file for each backup. The name of the archive file includes both the backend ID and the backup ID. The backup ID can be provided to the backup command, or an ID is generated from a current timestamp. Each backup can be optionally compressed, encrypted, hashed or signed. A backup taken on one system can be restored on another system. This tool features both an offline mode of operation as well as the ability to schedule an operation to run within the directory 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.
`base64`	Encode raw data using the base64 algorithm or decode base64-encoded data back to its raw representation.
`check-replication-domains`	Scan the `replicationChanges` database for all known replication domains and identify any obsolete replicas still listed as part of a topology. Run this tool before upgrading replicated topologies to help avoid lockdown mode. Learn more about Discovering obsolete replicas.
`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. Information collected can include configuration files, server monitor entries, portions of log files, JVM thread stack dumps, system metrics, and other data that might be helpful in diagnosing problems, understanding server performance, or otherwise assisting with support requests. Although the tool will do its best to obscure or omit sensitive data, and the entire archive can be encrypted if you desire, you might want to review the resulting support data archive to ensure verify its contents. Further, the archive will include a summary of any potential problems or concerns that are identified in the course of collecting the support data.
`compare-ldap-schemas`	Compares the schemas of two LDAP servers to identify schema elements that might be present in one but not the other, or elements that might be present in both servers but have differences between them.

`config-diff`	Compares directory server configurations and produces a `dsconfig` batch file needed to bring the source inline with the target. Its uses include comparing multiple servers for configuration differences, producing a batch file to reconfigure a server from scratch from the out-of-the-box configuration, and comparing a local server against an expected configuration. Both the source and the target configurations can be retrieved over LDAP, accessed from the local server's file system, extracted from a specific file, or retrieved from every server in a configuration server group. Also, with the exception of accessing a configuration from a specific file, the source and/or target configurations can be compared as they existed at any point in the past, including the baseline, pre-installation configuration. Some configuration differences (those that will always differ between instances, like instance-name) are excluded by default to reduce the amount of spurious output, but these can be included by specifying the `--includeExpectedDifferences` command. Further differences can be excluded with the `--exclude` option. This tool attempts to generate a batch file that can be applied to the source server without any errors. However, there are some edge case configurations that the tool is not sophisticated enough to handle. For example, it cannot handle two peer configuration objects that would require swapping values for a property (for example, evaluation-order-index) that must be unique within the server. It will still generate a `dsconfig` batch file that includes these changes, but they might be rejected by the server. In these rare cases, the batch file can be hand edited so that it can be applied to a running server or it can be applied with the server shut down using `dsconfig --offline`.
`create-rc-script`	Create a Run Control (RC) script that you can use to start, stop, and restart the PingDirectory server on UNIX-based systems.
`create-systemd-script`	Create a systemd script to start and stop the PingDirectory server on Linux-based systems.
`dbtest`	Inspect the contents of the PingDirectory 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. Each local DB backend has a root container, identified by the backend ID. Each root container has an entry container for each base distinguished name (DN) in the backend. Each entry container has several database containers that store entries, attribute indexes and system indexes. The `dbtest` command allows the contents of root containers, entry containers and database containers to be inspected.
`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 PingDirectory server configuration. This utility offers three primary modes of operation, the interactive mode, the non-interactive mode, and the batch mode: • The interactive mode supports viewing and editing the configuration using an intuitive, menu driven environment. Running `dsconfig` in interactive provides a user-friendly, menu-driven interface for accessing and configuring the server. To start `dsconfig` in interactive command-line mode, simply invoke the `dsconfig` shell script or batch file without any arguments. • The non-interactive mode provides a simple way to make arbitrary changes to the PingDirectory server by invoking it on the command-line. If you want to use administrative scripts to automate the configuration process, then run the `dsconfig` command in non-interactive mode. • The batch mode reads multiple `dsconfig` invocations from a file and executes them sequentially. The batch file provides advantages over standard scripting in that it minimizes LDAP connections and JVM invocations that normally occur with each `dsconfig` call. You can view the `logs/config-audit.log` file to review the configuration changes made to the PingDirectory server and to use them in the batch file.
`dsjavaproperties`	Configure the JVM options used to run the PingDirectory server and its associated tools. The options managed by this tool are stored in `config/java.properties`. Typically, you should not edit that file directly. Instead, run the tool specifying `--jvmTuningParameter` arguments to customize JVM options appropriate for this system. Note that the changes will only apply to this PingDirectory server installation. No modifications will be made to your environment variables. Memory and other settings for the JVM tools, including the `start-server` tool, can be tuned during initialization by specifying one or more instances of the `--jvmTuningParameter` option when invoking this tool. Supported values are as follows: • `NONE`: Explicitly specify no parameters. • `AGGRESSIVE`: This system is dedicated to running only this server. The amount of memory allocated to this server will be computed accordingly. • `SEMI_AGGRESSIVE`: This system is shared by multiple server processes. The amount of memory allocated to this server will be computed accordingly. If no parameters are specified, the parameters specified by the previous invocation of this tool or setup will be used. Use the `NONE` option to explicitly specify no parameters.

`dsreplication`	Manage data replication between two or more PingDirectory server instances. For replication to work, you must first to enable replication using the **`enable`** subcommand. Then, you initialize the contents of one of the servers with the contents of the other using the **`initialize`** subcommand.
`dump-dns`	Obtain a listing of all of the DNs for all entries below a specified base DN in the PingDirectory 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, LDAP Data Interchange Format (LDIF) exports, and log files generated by the server.
`encryption-settings`	Manage the server encryption settings database. More information about the cipher algorithms and transformations available for use can be found in the Java Cryptography Architecture Reference Guide, as well as the Standard Algorithm Name Documentation for your chosen JDK implementation used by this server.
`enter-lockdown-mode`	Request that the PingDirectory server enter lockdown mode, during which it only processes operations requested by users holding the `lockdown-mode` privilege. While in lockdown mode, the PingDirectory server rejects all requests from users that do not hold the `lockdown-mode` privilege.
`export-ldif`	Export data from the PingDirectory server backend in LDIF form. To export data from a remote PingDirectory server, the server must be running and connection parameters must be supplied. You can specify options to include or exclude specific attributes and branches of the tree, and to include or exclude entries matching a given filter. The data can be appended to an existing file instead of overwriting it, and the output can be optionally compressed. This tool features both an offline mode of operation as well as the ability to schedule an operation to run within the PingDirectory 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.

`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. Include the following arguments to configure the output of the tool: **`--filter`** The export only includes entries matching the specified filter. **`--exportNonReversiblePasswords`** The export includes entries with non-reversible passwords. **`--exportEntriesWithoutPasswords`** The export includes entries that don't have passwords. **`--includeAdditionalAttribute`** The exported entries include the specified attribute(s) in addition to the user's DN and password. You can use `"*"` to include all user attributes and `"+"` to include all operational attributes. **`--includeVirtualAttributes`** The exported entries include the values of virtual attributes in addition to real attributes. This tool can only be used over a secure connection and when authenticated as a user with the `permit-export-reversible-passwords` privilege. The server encrypts the output using a key generated from either a user-supplied passphrase or an encryption settings definition.

`extract-data-recovery-log-changes`	Extracts changes matching a given set of criteria from a PingDirectory 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). This tool is designed to be used in conjunction with the server's data recovery log files (in the `logs/data-recovery` directory). It can be used in conjunction with other audit log files, but for best results, the logger should be configured to operate in reversible form, to include the requester DN and IP address, and to include information about any intermediate client control that might have been provided in the request. This tool must not be used with a log file that the server can update while the tool is running, or that can have some content stored in an unwritten buffer (which is especially likely if the log is compressed or encrypted). To use this tool with the server online, you should only specify a log file that has already been rotated to ensure that no more writes will be made to that file. If necessary, use the `rotate-log` tool to force the current active file to be rotated. To use this tool to revert an inappropriate set of changes, run it with `--direction revert` and an additional set of arguments that identify which changes should be reverted (for example, based on the address of the client, the authorization DN of the requester, the time frame in which the changes were applied, etc.). To use this tool to replay changes that were previously applied (for example, after restoring an old backup or importing an old LDIF file), run it with `--direction replay` and an appropriate set of arguments to select the desired set of changes. Also, make sure to use `dsreplication pre-external-initialization` before performing the restore or import and applying the changes, and then use `dsreplication post-external-initialization` after the changes have been applied. See the PingDirectory Server Administration Guide for additional information. This tool will extract changes from the selected log file (and any previously rotated files, unless the `--doNotFollowRotationChain` argument is provided) and output them in LDIF change format. If the `--outputFile` argument is provided, then the changes will be written to that file; otherwise, they will be written to standard output. If changes are to be written to a file, then the output will be compressed if the input files were compressed (unless the `--doNotCompressOutput` argument was provided), and the output will be encrypted if the input files were encrypted (unless the `--doNotEncryptOutput` argument was provided). You might want to first run the tool without specifying an output file so that you can verify that the selected changes are correct. Once you are certain that the appropriate changes have been selected, you can use a tool like `ldapmodify` or `parallel-update` to apply those changes to the server.
`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 which 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. That is, it 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 a PingDirectory server backend. Connection parameters are not required when importing to a local PingDirectory server that is not running. However, connection parameters are required if the server is remote, or if it is running locally and it is inconvenient to have to stop it for the import. You can use the options to include or exclude specific attributes and branches of the tree, and to include or exclude entries matching a given filter. The input file can be compressed. This tool features both an offline mode of operation as well as the ability to schedule an operation to run within the PingDirectory 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.
`indent-ldap-filter`	Parse a provided LDAP filter string and display it a multi-line form that makes it easier to understand its hierarchy and embedded components. If possible, the tool 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.** **The `ldap-diff` tool outputs the difference between data stored in two LDAP servers into an LDIF file. This file could be used with the `ldapmodify` command to bring the source directory server in sync with the target directory server. The specific entries to compare can be controlled with the `--searchFilter` option. In addition, only a subset of attributes can be compared by listing those attributes as trailing arguments of the command. Specific attributes can also be excluded by prepending a `^` character to the attribute. On Windows operating systems, excluded attributes must be quoted. For example: `"^attrToExclude"`. When retrieving entries from a PingDirectory server, the `@objectClassName` notation can be used to compare only attributes that are defined for a given objectClass.** **This command can be used on servers actively being modified, without reporting false positives caused by replication delays, by checking differing entries multiple times. By default, it will re-check each differing entry twice, pausing two seconds between checks. These settings can be configured with the `--numPasses` and `--secondsBetweenPass` options. The output is formatted so that delete operations come first, modify operations come next, and add operations come last. This gives the best chance that the resulting output file can be used to bring the source server into sync with the target server without causing any conflicts. This takes into account attribute uniqueness constraints as well as the requirement that child entries be deleted before parents and parents be added before children.** **The directory 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 PingDirectory 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` **For servers from other vendors, consult their documentation for configuring the proper privileges.** **The `ldap-diff` tool tries to make efficient use of memory, but it must store the DNs of all entries in memory. For directories that contain tens 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 `ldap-diff` can be customized by editing the `ldap-diff.java-args` parameter in the `config/java.properties` file and running the `dsjavaproperties` command.**

`ldap-result-code`	Display and query LDAP result codes. This tool can be used to list all known defined LDAP result codes, retrieve the name of the result code with a given integer value, or search for all result codes with names containing a given substring. At most, one of the `--list`, `--int-value`, and `--search` arguments can be provided. If none of them is provided, then the `--list` option will be chosen by default.
`ldapcompare`	Perform compare operations in an LDAP directory server. Compare operations can be used to efficiently determine whether a specified entry has a given value. The exit code for this tool will indicate whether processing was successful or unsuccessful, and to provide a basic indication of the reason for unsuccessful attempts. By default, it will use an exit code of zero (which corresponds to the LDAP 'success' result) if all compare operations completed with a result code of either 'compare false' or 'compare true' (integer values 5 and 6, respectively), but if the `--useCompareResultCodeAsExitCode` argument is provided, only one compare assertion is performed, and it yields an exit code of 'compare false' or 'compare true', then the numeric value for that result code will be used as the exit code. If any error occurs during processing, then the exit code will be a nonzero value that reflects the first error result that was encountered. The attribute type and assertion value to use for the compare operations will typically be provided as the first unnamed trailing argument provided on the command line. It should be formatted with the name or OID of the target attribute type followed by a single colon and the string representation of the assertion value. Alternatively, the attribute name or OID can be followed by two colons and the base64-encoded representation of the assertion value, or it can be followed by a colon and a less-than symbol to indicate that the assertion value should be read from a file (in which case the exact bytes of the file, including line breaks, will be used as the assertion value). The DNs of the entries to compare can either be provided on the command line as additional unnamed trailing arguments after the provided attribute-value assertion, or they can be read from a file whose path is provided using the `--dnFile` argument. If the attribute-value assertion is provided on the command line as an unnamed trailing argument, then the same assertion will be performed for all operations. If multiple types of assertions should be performed, then you can use the `--assertionFile` argument to specify the path to a file containing both attribute-value assertions and entry DNs.
`ldapdelete`	Delete one or more entries from an LDAP directory server. You can provide the DNs of the entries to delete using named arguments, trailing arguments, a file, or 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. Unless the password change method is explicitly specified (using the `--passwordChangeMethod` argument), this tool will attempt to automatically determine which method is the most appropriate for the target server using information provided in the server's root DSE. If the server advertises support for the password modify extended operation, then that method will be used. If it appears to be an Active Directory server, then an Active Directory-specific password-change method will be selected, using a regular LDAP modify operation to update the `unicodePwd` attribute with a specially encoded value. Otherwise, a regular LDAP modify operation will be used to update the value of a specified password attribute. The new password to be set for the user can be specified in one of several ways. It can be directly provided on the command line, read from a specified file, interactively prompted from the user, or automatically generated by this tool. If the new password is not specified using any of those methods, and if the password is to be updated using the password modify extended operation, then the new password field of the request will be left blank so that the server generates a new password for the user and includes it in the response to the client. If no new password is specified and some other password change method is selected, then the tool will exit with an error. The current password for the user can also be specified. This is optional, although some servers might require a user to provide their current password when setting a new one. If a current password is provided (whether given as a command-line argument, read from a specified file, or interactively requested from the user), and if a regular LDAP modify operation is used to change the password, then the resulting modify request will include a deletion of the current value and an addition of the new value. If no current password is provided, then the modify request will replace any existing password(s) with the new value.
`ldapsearch`	Process one or more searches in an LDAP directory server. The criteria for the search request can be specified in several different ways, including providing all of the details directly using command-line arguments, providing all of the arguments except the filter using command-line arguments and specifying a file that holds the filters to use, or specifying a file that includes a set of LDAP URLs with the base DN, scope, filter, and attributes to return.

`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. This tool works best with small LDIF files because it reads the entire contents of the source and target LDIF files into memory so they can be quickly compared. If you encounter an out-of-memory error while running the tool, you might need to increase the amount of memory available to the JVM used to invoke it. The amount of memory available to the JVM can be customized by invoking the JVM with the `-Xms` and `-Xmx` arguments, which specify the initial and maximum amounts of memory that it can use, respectively. These arguments should be immediately followed, without any intervening space, by an integer and a unit to specify the amount of memory to be used. The unit can be either `m` to indicate that the size is in megabytes, or `g` to indicate that it is in gigabytes. For example, `-Xms512m` indicates that the JVM should be given an initial heap size of 512 megabytes, while `-Xmx2g` indicates that it should be given a maximum heap size of two gigabytes. When invoking the **`ldif-diff`** tool included in the installation of a Ping Identity server product, you can edit the `config/java.properties` file to specify the arguments to use when invoking the JVM. After modifying the file, run the **`dsjavaproperties`** tool to ensure that those changes will be used for subsequent tool invocations.
`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`** command cannot read the changes to apply from standard input. All of the change records will be read into memory before processing begins, so it is important to ensure that the tool is given enough memory to hold those change records. However, it will only operate on a single source entry at a time, so the size of the source LDIF file does not significantly impact the amount of memory that the tool requires. Note that the tool will attempt to correctly handle multiple changes affecting the same entry. However, because it only operates on one entry at a time, it cannot always behave in exactly the same way as if it were applying the changes over LDAP to a server populated with the source LDIF file. For example, it is not possible to reject an attempt to delete an entry that has subordinates, so any delete will be treated as a subtree delete. Further, not all types of modify DN change records are supported. In particular, modify DN change records are not permitted if they target any entry that has been targeted by a previous change record (for example, renaming an entry that was created by a previous add change record). Finally, it cannot perform other types of validation, like ensuring that all of the necessary superior entries exist when adding a new entry, or ensuring that a modify DN will not introduce a conflict with an existing entry.
`ldifsearch`	Search one or more LDIF files to identify entries matching a given set of criteria.

`leave-lockdown-mode`	Request that the PingDirectory server leave lockdown mode and resume normal operation. While in lockdown mode, the PingDirectory server rejects all requests from users that do not hold the lockdown-mode privilege. Note that the PingDirectory server can place itself in lockdown mode under certain conditions (for example, if it detects a security problem like a malformed access control rule that might have otherwise resulted in exposure of sensitive data).
`list-backends`	List the backends and base DNs configured in the PingDirectory server.
`load-ldap-schema-file`	Loads 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. See the server's `config/MakeLDIF` directory for example template files. 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 will be 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, PKCS #12, PKCS #11, or BCFKS key store.
`manage-extension`	Install or update PingDirectory server extension bundles. An extension bundle is a package of extension(s) that utilize the Server SDK to extend the functionality of the PingDirectory server. Extension bundles are installed from a zip archive or file system directory. The server will be restarted if running to activate the extension(s).
`manage-profile`	Generate, compare, install, and replace server profiles. Server profiles define a format for the configuration of a server, including `dsconfig`, initial DIT, setup arguments, server SDK extensions, and other files. These are combined into one concrete structure. This tool provides subcommands that can be used to generate a new profile from an existing server, to set up a new server, and to replace an existing server's profile with a different profile. A template server profile file structure can be found in the resource/ directory.
`manage-tasks`	Access information about pending, running, and completed tasks scheduled in the PingDirectory server.

`manage-topology`	Manage the topology registry. The topology registry is a branch of the configuration DIT (`cn=Topology,cn=config`). It stores all metadata about server instances, including their instance and listener certificates, secret keys, server groups and administrative user accounts. In addition, it also stores information about the replication topology (replication server ID and replication domain ID) when replication is enabled among servers in a Directory topology. Last but not least, it stores the license key required to install the server. Changes to the topology registry on one server are automatically mirrored to other servers in the topology. The **`dsconfig`** tool, configuration API, or the management console can be used to make changes to the topology registry. This tool allows some additional capability such as exporting the contents of the registry as a JSON file.
`migrate-ldap-schema`	Migrate schema information from an existing LDAP server into a PingDirectory server instance. This tool can be used to migrate schema information from an existing LDAP server into this PingDirectory server instance. The source server can be any standards-compliant LDAPv3 server. All attribute type and object class definitions, which are contained in the source LDAP server but not in the target PingDirectory server instance, will be either added to the target instance or written to a schema file.
`migrate-sun-ds-config`	Update an instance of the PingDirectory server to match the configuration of an existing Sun Java System Directory Server 5.x, 6.x, or 7.x. This tool can be used to compare the configuration of Sun Java System Directory Server 5.x, 6.x, or 7.x and PingDirectory server instances in order to identify any differences and update the PingDirectory server configuration to more closely match that of the Sun server instance.
`modrate`	Perform repeated modifications against an LDAP directory server.
`move-subtree`	Move all entries in a specified subtree from one server to another.
`oid-lookup`	Search the OID registry to retrieve information about items that match a given OID or name. The string to use to search the OID registry should be provided as an unnamed trailing argument. All items in the OID registry will be examined, and any items that contain the provided string in its OID, name, type, origin, or URL will be matched. If no search string is provided, the entire OID registry will be displayed.

`parallel-update`	Use multiple concurrent threads to apply a set of add, delete, modify, and modify DN operations read from an LDIF file. As with other tools like **`ldapmodify`**, changes in the LDIF file to be processed should be ordered such that if there are any dependencies between changes (for example, if one add change record creates a parent entry and another add change record creates a child of that parent), prerequisite changes come before the changes that depend on them. When this tool is preparing to process a change, it will determine whether the new change depends on any other changes that are currently in progress, and if so, will delay processing that change until its dependencies have been satisfied. If a change does not depend on any other changes that are currently being processed, then it can be processed in parallel with those changes. This tool will keep track of any changes that fail in a way that indicates they succeed if re-tried later (for example, an attempt to add an entry that fails because its parent does not exist, but its parent can be created later in the set of LDIF changes), and can optionally re-try those changes after processing is complete. Any changes that are not retried, as well as changes that still fail after the retry attempts, will be written to a rejects file with information about the reason for the failure so that an administrator can take any necessary further action upon them.
`populate-composed-attribute-values`	Populate entries in one or more backends with attribute values generated by one or more composed attribute plugins. This tool uses the configuration from a specified set of composed attribute plugin instances to identify which entries to update and what changes to apply. It can be used as an alternative to exporting the data to LDIF and re-importing to ensure that existing entries have an appropriate set of composed attribute values. This tool schedules an operation to run within the PingDirectory server's process. LDAP connection options must be supplied 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.
`profile-viewer`	View information in data files captured by the PingDirectory server profiler. Profiler data files are generated by the Profiler plugin. To create these data files, set the profile-action attribute of the Profiler configuration object to 'start' to begin collection. Set the `profile-action` attribute to `stop` to end collection and have the plugin write the file to `logs/profile`.{timestamp}.

`re-encode-entries`	Re-encode all or a specified portion of the entries in a local DB backend. This tool can be used to initiate a task that will cause a local DB backend to re-encode all or a specified subset of the entries that it contains. The contents of the entries will not be altered, but this provides a useful mechanism for applying significant changes to the way that entries are actually stored in the backend (for example, to apply encoding changes if a feature like data encryption or uncached attributes or entries is enabled). This tool schedules an operation to run within the PingDirectory server's process. LDAP connection options must be supplied 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.
`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. An index must be rebuilt if the database already contains data when the index is configured. The backend containing the provided base DN must be a local DB backend. The types of indexes that can be rebuilt include attribute indexes, VLV indexes, and JSON field indexes. **Note** PingDirectory does not support the rebuilding of system indexes, regardless of whether the rebuild is attempted online or offline. If you need to rebuild a system index, you must export the backend to LDIF and re-import it. This tool features both an offline mode of operation as well as the ability to schedule an operation to run within the PingDirectory 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.
`register-yubikey-otp-device`	Registers a YubiKey OTP device with the PingDirectory 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. This tool schedules an operation to run within the PingDirectory server's process. LDAP connection options must be supplied 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.
`remove-attribute-type-from-schema`	Safely remove an attribute type definition from the server schema. The tool will perform an appropriate set of validation before actually removing the attribute type from the schema. The below conditions must be satisfied before the attribute type can be removed. • The requester must have the update-schema privilege. • The attribute type must not be referenced by any other schema element. • The attribute type must not be defined in any schema file that is included with the PingDirectory server. Only custom attribute types can be removed from the schema. • The attribute type must not be referenced in the server configuration (for example, it must not be indexed by any backend). • The attribute type must not be present in any entry that exists in any backend. This tool schedules an operation to run within the PingDirectory server's process. LDAP connection options must be supplied 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.
`remove-backup`	Safely remove a backup from the specified PingDirectory server backend. This tool deletes the specified backup archive and updates the backup descriptor accordingly. As an alternative to removing a specific backup, you can automatically remove backups outside of specified count or age criteria. The `--retainFullBackupCount` argument can be used to indicate that the specified number of full backups should be retained, and any other full backups in the directory are eligible to be removed. The `--retainFullBackupAge` argument can be used to indicate that any full backups older than the specified age are eligible to be removed.
`remove-defunct-server`	Remove a server from this server's topology. This tool will remove the specified server from the topology. In general, the `uninstall` tool should be used to remove a server from the topology. The `remove-defunct-server` tool should only be used if a prior attempt to uninstall a server was unsuccessful or the system where the server was installed is no longer available, leaving the server permanently inaccessible from the topology. If the defunct server is online and is able to reach other servers in the topology, running remove-defunct-server from it will cleanly remove it from the topology. If it cannot reach the other servers, then remove-defunct-server must also be run from one of the online servers.

`remove-object-class-from-schema`	Safely remove an object class definition from the server schema. The tool will perform an appropriate set of validation before actually removing the object class from the schema. The below conditions must be satisfied before the object class can be removed. • The requester must have the update-schema privilege. • The object class must not be referenced by any other schema element. • The object class must not be defined in any schema file that is included with the PingDirectory server. Only custom object classes can be removed from the schema. • The object class must not be referenced in the server configuration. • The object class must not be present in any entry that exists in any backend. This tool schedules an operation to run within the PingDirectory server's process. LDAP connection options must be supplied 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.
`replace-certificate`	Replace the listener certificate for this PingDirectory server instance.
`restore`	Restore a backup of a PingDirectory server backend. Only one backend can be restored at a time by the restore command. The PingDirectory server should be stopped unless task connection options are supplied for a running server. You can list the backups contained in a particular backend backup directory. A backup taken on one system can be restored on another system. This tool features both an offline mode of operation as well as the ability to schedule an operation to run within the PingDirectory 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.
`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. If the file argument is provided one or more times to specify the target log file paths, then only those log files will be rotated. If the file argument is not given, then the server will trigger rotation for all supported log files. You must have the `config-read` and `config-write` privileges to run this tool, and you must have the necessary access control rights to create and monitor entries in the task backend. This tool schedules an operation to run within the PingDirectory server's process. LDAP connection options must be supplied 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.
`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, several 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 • 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 the **`search-logs`** 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 PingDirectory server process.
`set-delegated-admin-aci`	Request that the PingDirectory server assign appropriate ACI for configured delegated administrators of the Delegated Admin API.
`setup`	Perform the initial setup for a server instance. This tool features both interactive and non-interactive modes for accepting the product license terms and initially configuring a server instance.
`start-server`	Start the PingDirectory server.
`status`	Display basic server information. This tool prints information about the server, such as version, connection handlers, and data sources. Some information might not be available if the server is not running, or if authentication credentials are missing or do not have sufficient privileges, or if the invoking user does not have sufficient file system access rights.
`stop-server`	Stop or restart the server. This tool is used to stop or restart the local instance of the server (by omitting LDAP connection options), or a remote server (by interacting with it over LDAP). In addition, this tool is used to schedule the server for shutdown at a later time using the server's task interface.
`subtree-accessibility`	List or update the set of subtree accessibility restrictions defined in the PingDirectory server.
`sum-file-sizes`	Calculate the sum of the sizes for a set of files. This tool is used to find the sum of the sizes of one or more files. If any of the files specified is a directory then it will be recursively processed.
`summarize-access-log`	Examine one or more access log files to display several metrics about operations processed within the server.
`sync-pipe-view`	PingDataSync only: Display the detailed configuration of a sync pipe or pipes in PingDataSync and all commands necessary to replicate a specified sync pipe. You can use the `sync-pipe-view` tool online with the PingDataSync server connection credentials or you can use the tool offline. The sync pipe information can be output in a text, CSV, JSON, or tab-delimited format and can be output to a file. The `sync-pipe-view` tool features both an interactive mode and a non-interactive mode.
`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 the PingDirectory server. This tool removes the entire server or individual server components from the file system. If this server is a member of a replication topology, you must first remove references to this server in the other servers using the `dsreplication disable` command.
`update`	Update a deployed server so its version matches the version of this package.
`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 PingDirectory server. Note that output generated by this tool will be LDIF, but each entry in the output will have exactly one ACI, so entries which 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. The backend containing the provided base DN must be a local DB backend. The types of indexes that can be verified include system indexes, attribute indexes and VLV indexes. Any errors found during verification are written to the output. The verification process is exhaustive and can take a long time.
`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

The PingDirectory 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 the PingDirectory server in multiple environments.

Each property takes the form of a name-value pair that defines predetermined values for a tool's options.

The PingDirectory server supports the following types of properties:

- Default properties that apply to all command-line utilities
- Tool-specific properties

Evaluation of command-line options and file options

You can specify options for a command-line tool on the command line, in a properties file, or both.

Options you specify on a tool's command line take priority over options in a properties file.

Consider the following scenarios.

Command-line options	Server uses ...
No command-line options	The options in the default `<server-root>/config/tools.properties` file.
Command-line options other than the `--propertiesFilePath <my-properties-file>` option	The command-line options, which take priority if the options are also in the `<server-root>/config/tools.properties` file. The file options for options that are only in the default `<server-root>/config/tools.properties` file.
Only the `--propertiesFilePath <my-properties-file>` option	The options in `<my-properties-file>`.
The `--propertiesFilePath <my-properties-file>` option and other command-line options	The command-line options, which take priority if the options are also in `<my-properties-file>`. The file options for options that are only in `<my-properties-file>`.
The `--noPropertiesFile` option and other command-line options	Only the options you specify on the command line, ignoring the default properties file.

Example

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
```

The server checks command-line options and file options to determine the options to use, as explained below.

- All options presented with the tool on the command line take precedence over any options in a properties file.

  In the following example, the command runs with the options specified on the command line (`--port` and `--baseDN`). With the `port` value both on the command line and in the properties file, the command-line value takes priority. 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=*)"
  ```

- If you specify the properties file using the `--propertiesFilePath` option and no other command-line options, the server uses only the options in the specified properties file:

```
$ bin/ldapsearch --propertiesFilePath bin/tools.properties \
  "(objectclass=*)"
```

- If you do not specify any command-line options, the server attempts to locate the default properties file in the following location:

```
<server-root>/config/tools.properties
```

By moving your `tools.properties` file from `<server-root>/bin` to `<server-root>/config`, you do not have to specify the `--propertiesFilePath` option. That change shortens the previous command to the following command:

```
$ bin/ldapsearch "(objectclass=*)"
```

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 Lightweight Directory Access Protocol (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.

dsconfig batch files">

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

The PingDirectory 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.**

PingDirectoryProxy Server Administration Guide

PingDirectoryProxy is a fast and scalable LDAPv3 gateway for the PingDirectory server. Administrators can configure the PingDirectoryProxy server architecture to control how client requests are routed to backend servers.

PingDirectory product documentation

Support

https://support.pingidentity.com

Overview of the PingDirectoryProxy features

This section gives an overview of PingDirectoryProxy features and components.

PingDirectoryProxy is a fast and scalable LDAPv3 gateway for the PingDirectory server. Administrators can configure the PingDirectoryProxy server architecture to control how client requests are routed to backend servers.

Overview of the PingDirectoryProxy server components and terminology

This section describes each PingDirectoryProxy server component in more detail.

The PingDirectoryProxy server provides the following components and functionalities for 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

About locations

You can assign a location to the PingDirectoryProxy server and each of the backend LDAP external servers for routing requests and failover response preferences.

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. These locations can determine how to route requests so that the server forwards requests to the PingDirectoryProxy server in the same data center over those in remote locations.

For example, a deployment consists of three data centers, one in New York, one in Chicago, and one in Los Angeles. In the New York data center, applications that 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.

Follow these guidelines for assigning locations in the PingDirectoryProxy server:

- If you have multiple data centers, assign a separate location for each one.
- In most environments, all PingDirectoryProxy server instances should have the same configuration except for the attribute that specifies the location of the server itself.

Learn more about Configuring locations.

About LDAP external servers

PingDirectoryProxy allows you to configure different types of LDAP external servers.

You can configure information about the directory server instances accessed by PingDirectoryProxy. The default configuration for each type is tuned to be the best possible configuration for each.

The 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 might allow only reads and others allow reads and writes so that the PingDirectoryProxy server can recognize this and accommodate it.

For more information about configuring LDAP external servers, see Configuring LDAP external servers.

About LDAP health checks

LDAP health checks provide information about the status and availability of LDAP external servers. Configure the PingDirectoryProxy server health checks that work best for your environment.

PingDirectoryProxy provides the following health checks:

Measure the response time for searches and examine the entry contents

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 PingDirectoryProxy 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 PingDirectory server indicates there is a problem, such as an index that must be rebuilt, then it flags itself as `degraded` or `unavailable`. When the PingDirectoryProxy server detects this, it stops sending requests to the server. The PingDirectoryProxy 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 PingDirectory server.

When the PingDirectoryProxy server is notified by the PingDirectory server of a new alert, it immediately retrieves the base `cn=monitor` entry of the PingDirectory server. If this entry has a value for the `unavailable-alert-type` attribute, then the PingDirectoryProxy server considers it unavailable. If this entry has a value for the `degraded-alert-type` attribute, then the PingDirectoryProxy server considers it `degraded`. Clients of the PingDirectoryProxy server can use a similar mechanism to detect and react when server flags itself as `degraded` or `unavailable`.

Monitor the busyness of the server

If a server becomes too busy, you can mark it as `degraded` or `unavailable` so that less heavily-loaded servers are preferred.

Results

The health check results contain the following server states:

`Available`

Completely accessible for use.

`Degraded`

The server can be used if necessary but has a condition which can 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 include a numeric score that has a value between 1 and 10. This score helps rank servers with the same state. For example, two servers are available and one has a score of 8 and the other a score of 7, you can configure the PingDirectoryProxy server to prefer the server with the higher score.

The results of health checks are made available to the load-balancing algorithms to help determine where to send requests. The PingDirectoryProxy server attempts to use servers with a state of `available` before trying servers with a state of `degraded`. It never attempts to use servers with a state of `unavailable`.

Some load-balancing algorithms also take the health check score into account, such as the health-weighted load-balancing algorithm that prefers servers with higher scores over those with lower scores. You should configure the algorithms that work best for you environment.

Frequency

The PingDirectoryProxy server periodically invokes health checks to monitor each LDAP external server and initiates 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. The PingDirectoryProxy server contains safeguards 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.

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.

You can find more information about configuring the properties of the external server in Configuring an external server using **`dsconfig`**.

Server states and search response times

In some cases, an LDAP health check defines different sets of criteria for promoting and demoting the state of a server. A `degraded` server might need to meet more stringent requirements to be reclassified as `available` than originally for it to be considered `degraded`.

If response time is used in the process of determining the health of a server, then the PingDirectoryProxy server might 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 helps avoid cases in which a server repeatedly transitions between the two states because it's operating near the threshold.

For example, the health check used to measure search response time is configured to mark any server as `degraded` when the search response time is greater than 1 second. You can configure that the response time must be less than 500 ms before the server is made available again so that the PingDirectoryProxy server doesn't flip back and forth between `available` and `degraded`.

You can find more information about configuring health checks in Configuring server health checks.

About load-balancing algorithms

Load-balancing algorithms determine which server in a set of similar servers to use to process a client request.

The PingDirectoryProxy server provides the following load-balancing algorithms:

Fewest operations

Forwards request to the backend server with the fewest operations currently in progress.

Single server

Sends requests to the same server and doesn't 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

Sends requests to a given server first. If that server fails, then the request sends to another specified server as specified in the ordered failover server list.

Learn more about Configuring load balancing.

Algorithm criteria

The algorithm takes the following criteria into account:

Location of the server

Servers in the same location as the PingDirectoryProxy server are preferred over those in alternate locations.

Health of the server

Servers that are `available` are preferred over those that are `degraded`. In some cases, the health check score can be used to further differentiate between servers with the same health check state.

Route requests consistency

Requests from a single client can be consistently routed to the same PingDirectoryProxy server instance to avoid problems such as propagation delay from replication.

Operation retries

Retries 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.

Proxy transformations

Proxy transformations are used to rewrite requests and responses as they pass through the PingDirectoryProxy server. Proxy data transformations are helpful for clients that use an old schema or that contain a hard-coded schema.

Use the PingDirectoryProxy server proxy transformations to:

- Provide a distinghuished name (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 PingDirectoryProxy 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.

- Alternatively, if a client tries to use the attribute `userID`, the PingDirectoryProxy server can map it to `uid` before sending the request on to the backend LDAP server. The PingDirectoryProxy server then remaps the response to `userID` when the value is returned.

- Suppress a specified attribute so that it is never returned to clients. Also, it can cause the server to reject requests which target that particular attribute.

- Prevent entries that match a given search filter from being returned to clients.

For more information about configuring proxy transformations, see Configuring proxy transformations.

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:

- Process the operation directly
- Forward the request to another server
- Hand off the request to another request processor

PingDirectoryProxy 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 Berkeley DB backend. For more information about the controls, see LDAP SDK for Java documentation.

PingDirectoryProxy provides the following types of request processors:

Proxying request processors

Forwards operations received by the PingDirectoryProxy server to other LDAP external servers.

Entry-balancing request processors

Splits 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

Perform ordered failover between other types of request processors, sometimes with different behavior for different types of operations.

Learn more about Configuring request processors.

About server affinity providers

The PingDirectoryProxy server supports the ability to forward a sequence of requests to the same external server if specific conditions are met, called server affinity.

Use the server affinity provider to establish an affinity to a particular backend server for certain operations.

The following provider configurations and server affinity methods are available in the PingDirectoryProxy server:

Client connection Server Affinity

Requests from the same client connection consistently route to the same backend server.

Client IP address Server Affinity

All requests coming from the same client system consistently route to the same backend server.

Bind DN Server Affinity

All requests from the same user consistently route 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 directory information tree (DIT) available to a client by associating a request processor with a base distinguished name (DN).

A 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 are involved in processing a request, such as searches that have a scope that is larger than the subtree view.

The subtree view includes using a single base DN to identify the portion of the DIT. They might have hierarchical relationships: for example, one subtree view is 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

PingDirectoryProxy maintains either one or two connection pools to the backend server, depending on the type of backend server you use.

PingDirectoryProxy maintains either one pool for all types of operations or two separate pools for processing bind and non-bind operations from clients. When PingDirectoryProxy establishes connections, it authenticates them using the authentication mechanism defined in the configuration of the external server.

These connections are re-used for all types of operations forwarded to the backend server. You can configure the bind distinguished name (DN) and password in the PingDirectoryProxy server.

When a client sends a bind request to the PingDirectoryProxy server, the server looks at the type of bind request that was sent:

- If the bind request is a SASL bind request, authentication is processed by the PingDirectoryProxy server itself and does not forward to the backend server, however, the PingDirectoryProxy server can 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, the PingDirectoryProxy server forwards the client request to the backend server so that it uses the credentials provided by the client.

Regardless of the authentication method the client uses, the PingDirectoryProxy server remembers the identity of the client after the authentication completes. For any subsequent requests sent by that client, the server uses 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 PingDirectoryProxy server configuration, the request processes through 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 contains 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 PingDirectoryProxy server
  - 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 can request. You can specify whether a particular type of operation is allowed for clients.

For some operation types, such as extended operations, you can allow only a particular subset of an operation type, such as a particular extended operation.

- A set of subtree views that define information about the parts of the directory information tree (DIT) the client can 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 is evaluated again to determine if it matches the same or a different client connection policy. The connection is terminated if it no longer matches any policy.

Learn more about Client connection policy configuration.

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. A global index is an in-memory cache used for quickly determining which set or sets of servers to use to process a request based on the entry distinguished names (DNs) and attribute values.

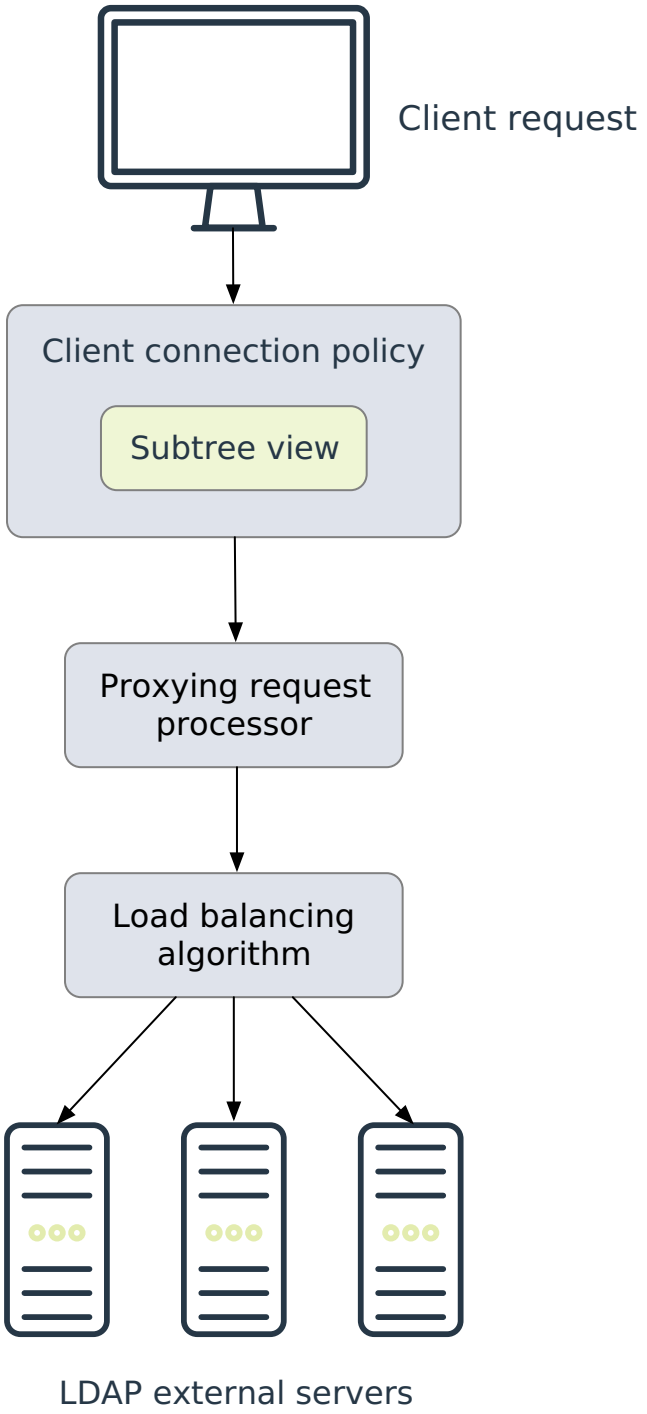
For information about configuring entry balancing, see Deploying an entry-balancing proxy configuration.

Server component architecture

This section provides an overview of the process flow between the PingDirectory server components for a simple proxy deployment and an entry-balancing deployment.

Architecture of a simple PingDirectory server deployment

A simple PingDirectory server deployment is illustrated in the following figure.



Processing steps

1. A client request is initiated.
2. The client connection policy processes the client request.
3. The client connection policy contains a subtree view, which defines the portion of the directory information tree (DIT) available to clients and determines if the DIT is available.
4. After the PingDirectory server determines that the DIT is available, it passes the request to the request processor, which defines the logic for processing the request.

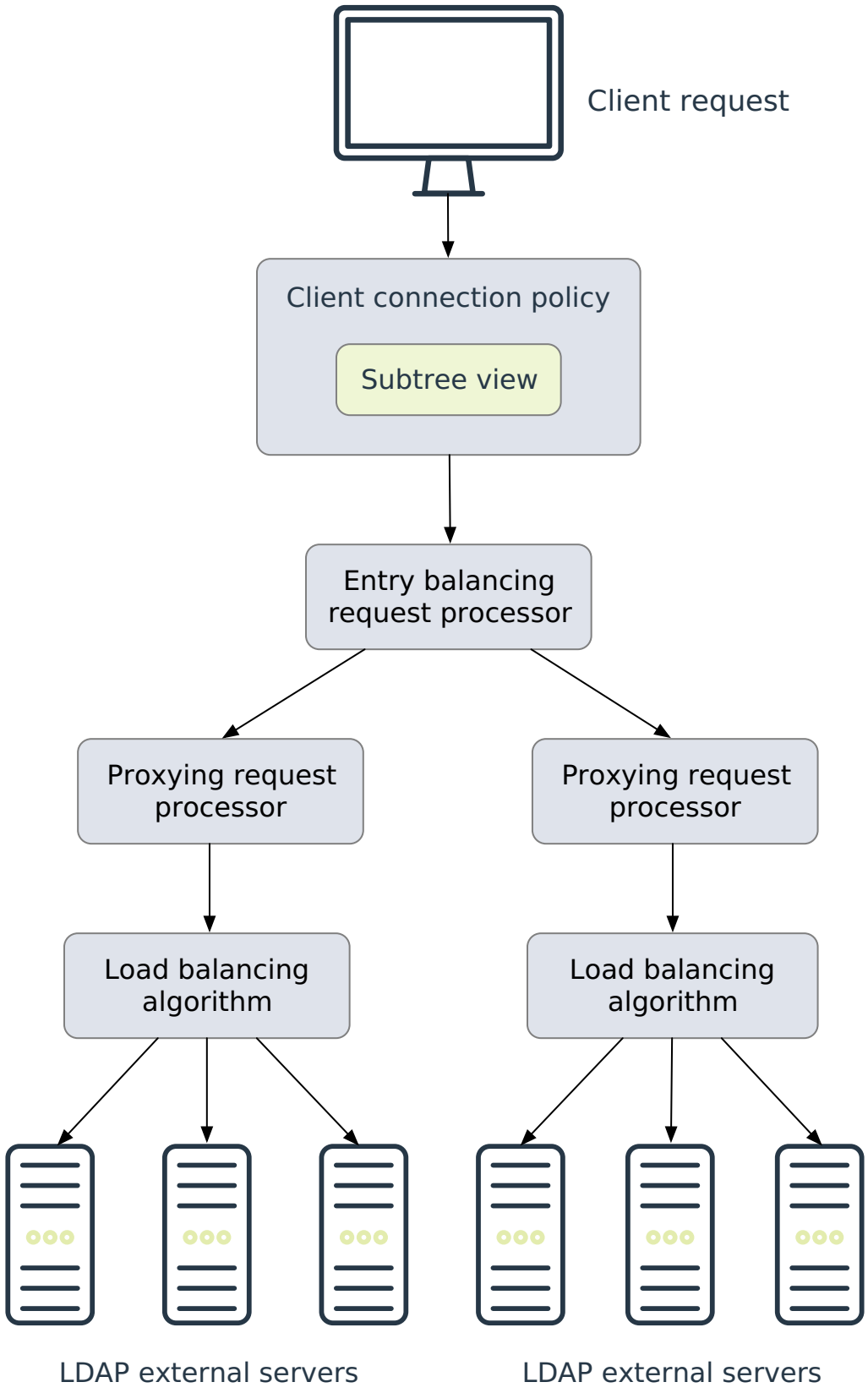
5. The request processor passes the request to a load-balancing algorithm, which determines the server in a set of servers responsible for handling the request.
6. The request is passed to the LDAP external server.

> **Note**
>
> 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 can be used by the load-balancing algorithm to determine how to route requests.

Architecture of an entry-balancing PingDirectory server deployment

Entry balancing is used when the data set is 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 can 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.

The following figure shows how a client request is treated in an entry-balancing PingDirectory server deployment.



1. A client request is initiated.

2. The client connection policy processes the client request.

3. The client connection policy contains a subtree view, which determines how the PingDirectoryProxy server communicates with a set of clients. It contains a subtree view that represents the base distinguished name (DN) for the entire deployment. The data set splits beneath this base DN.
4. The PingDirectory server passes the request 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.
5. The entry-balancing request processor has beneath it multiple proxying request processors that handle multiple unique sets of data.
6. The proxying request processors passes the request to a load-balancing algorithm, which determines which LDAP external server should handle the request.

   > **Note**
   >
   > As with a simple proxy deployment, this LDAP external server contains properties that define the server's location and the health checks that determine if the server is functioning properly.
7. The request is passed to the LDAP external server.

You can find more information about entry-balancing replication in Overview of replication in an entry-balancing environment.

PingDirectoryProxy server configuration overview

The following topic provides an overview of the component configurations for your PingDirectoryProxy server deployment.

Configuring 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 are configured as two locations in the PingDirectoryProxy server. Each location is associated with a name and an ordered list of failover locations to use if none of the servers in the preferred location are available.

Configuring the PingDirectoryProxy server location

You update the configuration to specify the location of the PingDirectoryProxyserver instance.

Configuring health checks for the LDAP external servers

You configure at what point the PingDirectoryProxy 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

You define each of the external directory servers, including the server type. You can configure Ping Identity directory servers, Java System Directory Servers, or generic LDAP servers. You also assign the server-specific health checks configured previously.

Configuring the load-balancing algorithm

You configure the load-balancing algorithm that the PingDirectoryProxy server uses to determine which server in a set of similar servers to use to process a client request. The PingDirectoryProxy server provides default algorithms. You can create new algorithms either using an existing algorithm as a template or creating one from scratch.

Configuring the proxying request processor

You configure proxying request processors that forward operations received by the PingDirectoryProxy server to other LDAP external servers.

Configuring subtree views

A subtree view defines the portion of the directory information tree (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 the PingDirectoryProxy server manages different client connections. The client connection policy can be used to control the types of operations that a client might perform and the portion of the DIT that the client can access. Restrictions configured in a client connection policy take precedence over any capabilities granted by access control or privileges.

Running the server as a Microsoft Windows service

The server can run as a Windows service, which enables you to sign 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 can't register a server while it's 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.

> **Note**
>
> Command-line arguments for the `start-server.bat` and `stop-server.bat` scripts aren't 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 can't 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

1. These configurations can be changed in the `<server-root>\config\wrapper.conf` file.

Configuring the PingDirectoryProxy server

After you have set up the PingDirectoryProxy server, you can manage your deployment using the configuration framework and management tools.

About the configuration tools

You can access and modify the server configuration in two ways.

Admin console

The server provides an admin console for graphical server management and monitoring. The console functions are equivalent to the `dsconfig` tool for viewing or editing configurations.

All configuration changes using the admin console are recorded in `logs/config-audit.log`, which also has the equivalent reversion commands if you need to undo a configuration.

`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
- 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 PingDirectoryProxy server. We strongly recommend that you use the `create-initial-proxy-config` tool for your initial server configuration. This tool prompts you for basic information about your topology, including external servers, their locations, and credentials for communicating with them. When the configuration is complete, the tool writes the configuration to a `dsconfig` batch file and allows you to apply the configuration to the local PingDirectoryProxy 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 caused by reimporting data.
- All servers support the same type of communication security.
- All external servers are any combination of PingDirectory 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 PingDirectoryProxy server's `logs` directory.

You can only run the `create-initial-proxy-config` tool once for the initial configuration of each PingDirectoryProxy server instance. To tune your configuration, use the `dsconfig` tool. When installing a second PingDirectoryProxy server, it will not be necessary to run the `create-initial-proxy-config` tool again, because the PingDirectoryProxy 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 PingDirectoryProxy server deployment as well as an entry balancing configuration.

Configuring a standard PingDirectoryProxy server deployment

Install a standard PingDirectoryProxy server deployment using the `create-initial-proxy-config` tool.

About this task

> **Note**
>
> Remember that you deploy PingDirectoryProxy servers 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, or run it manually at the command line from the server root directory, `<server-root>/PingDirectoryProxy`.

```
$ ./bin/create-initial-proxy-config
```

2. If the servers do not meet the displayed configuration requirements, 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, 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 distinguished name (DN) for the PingDirectoryProxy server user account, and then enter and confirm the password for this account.

```
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':
```

> **Note**
>
> You should not use `cn=Directory Manager` account for communication between the PingDirectoryProxy server and the PingDirectory server.
> For security reasons, the account used to communicate between the PingDirectoryProxy server and the PingDirectory server should not be directly accessible by clients accessing the PingDirectoryProxy server.
> For more information about this account, see Configuring LDAP external servers.

4. Specify whether to use secure communication with the PingDirectory 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. Enter the base DNs of the PingDirectory server instances that will be accessed through the PingDirectoryProxy server. Press Enter when you have finished specifying the DNs.

```
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]:
```

   The PingDirectoryProxy server will create subtree views using each base DN to define portions of the external servers' directory information tree (DIT) available for client access. You can specify more than one base DN.

6. Specify whether the entries under your defined subtree view will be split across multiple servers in an entry balanced deployment.

   Press Enter to accept the default setting of no.

7. Define a location for your server, such as the name of your data center or the city where the server is located.

```
Enter a location name or choose a menu item: east
```

8. (Optional) If you defined more than one location, specify the location that contains the PingDirectoryProxy 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 host:port used by the LDAP external servers.

```
Enter a host:port or choose a menu item [localhost:389]: ldap-east-01.example.com:389
```

> **Note**
>
> If you have specified more than one location, you will complete this process for each location.

10. 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 does not already 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. When finished, press Enter to finish configuring the location.

13. Review the configuration summary. After 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. Enter `yes` to apply the configuration changes locally to the PingDirectoryProxy server.

```
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]:
```

> **Note**
>
> If you have any Server SDK extensions, be sure to run the **`manage-extension`** tool first, then press Enter to apply the changes to the PingDirectoryProxy server.

Alternatively, you can quit and run the `dsconfig` batch file at a later time.

After the changes have been applied, you cannot use the `create-initial-proxy-config` tool to configure this PingDirectoryProxy server again. Use the `dsconfig` tool to modify your configuration instead.

Result

If you open the generated `proxy-cfg.txt` file or the `logs/config-audit.log` file, you see that a configuration element hierarchy has been created, listing 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 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.

To view the options for the dsconfig tool, change to the `PingDirectory/bin` directory, and enter `./dsconfig --help`. Example output is shown below.

```
./dsconfig --help

View and edit the Directory Server configuration.
This utility offers three primary modes of operation, the interactive mode, the non-interactive mode
and batch mode.  The interactive mode supports viewing and editing the configuration via an intuitive,
menu driven environment.  Running dsconfig in interactive command-line mode provides a user-friendly,
menu-driven interface for accessing and configuring the server. To start dsconfig in interactive
command-line mode, simply invoke the dsconfig shell script or batch file without any arguments.

The dsconfig non-interactive command-line mode provides a simple way to make arbitrary changes to the
Ping Identity Directory Server by invoking it on the command-line. If you want to use administrative
scripts to automate the configuration process, then run the dsconfig command in non-interactive mode.

The dsconfig tool provides a batching mechanism that reads multiple dsconfig invocations from a file
and executes them sequentially.  The batch file provides advantages over standard scripting in that it
minimizes LDAP connections and JVM invocations that normally occur with each dsconfig call.  You can
view the logs/config-audit.log file to review the configuration changes made to the Ping Identity
Directory Server and use them in the batch file.

Subcommands

  See the Usage section for instructions on viewing the list of supported subcommands.

Usage:  dsconfig  {options}
        where {options} include:

    --applyChangeTo [server-group|server-group-force|single-server]
        Controls whether changes apply to a single server or all servers in the configuration server group
    --offline
        Interact with the local configuration while the server is offline.  Not for use while the server
        is running
    -r, --reason {reason}
        A string describing the reason for the configuration change
    --help-classifications
        Display subcommands relating to connection and operation classification
    --help-core-server
        Display subcommands relating to core
    --help-database
        Display subcommands relating to backends, indexing, and caching
    --help-logging
        Display subcommands relating to logging, monitoring, and notifications
    --help-replication
        Display subcommands relating to replication
    --help-security
        Display subcommands relating to security and authorization
    --help-topology
        Display subcommands relating to topology
    --help-user-management
        Display subcommands relating to authentication and password management
    --help-web
        Display subcommands relating to web services and applications
    --help-subcommands
        Display all subcommands
```

```
  Configuration Options

    --advanced
        Allow the configuration of advanced components and properties

  LDAP Connection Options

    -Z, --useSSL
        Use SSL for secure communication with the server
    -q, --useStartTLS
        Use StartTLS to secure communication with the server
    --useNoSecurity
        Use no security when communicating with the server
    -h, --hostname {host}  [Default: localhost]
        Directory Server hostname or IP address
    -p, --port {port}  [Default: 389]
        Directory Server port number
    -D, --bindDN {bindDN}  [Default: cn=Directory Manager]
        DN used to bind to the server
    -w, --bindPassword {bindPassword}
        Password used to bind to the server
    -j, --bindPasswordFile {bindPasswordFile}
        Bind password file
    -o, --saslOption {name=value}
        SASL bind options (can be specified multiple times)
    -X, --trustAll
        Trust all server SSL certificates
    -P, --trustStorePath {truststorePath}  [Default: /Users/rowannabobo/Desktop/PingDirectory_9.2/config/
truststore]
        Certificate truststore path
    -T, --trustStorePassword {truststorePassword}
        Certificate truststore PIN
    -U, --trustStorePasswordFile {path}
        Certificate truststore PIN file
    --trustStoreFormat {trustStoreFormat}
        Certificate truststore format
    -K, --keyStorePath {keystorePath}
        Certificate keystore path
    -W, --keyStorePassword {keystorePassword}
        Certificate keystore PIN
    -u, --keyStorePasswordFile {keystorePasswordFile}
        Certificate keystore PIN file
    --keyStoreFormat {keyStoreFormat}
        Certificate keystore format
    -N, --certNickname {nickname}
        Nickname of the certificate for SSL client authentication

  Utility Input/Output Options

    -v, --verbose
        Use verbose mode
    -Q, --quiet
        Use quiet mode
    -n, --no-prompt
        Use non-interactive mode.  If data in the command is missing, you will not be prompted and the
```

```
        tool will fail
    -F, --batch-file {batchFilePath}
        Path to a file containing a sequence of dsconfig commands to run
    --batch-continue-on-error
        Force the execution of all commands in the batch file on the server even if prevalidation fails.
        Execution will also continue even if one of the commands fails.
        Please note that commands affecting multiple servers can still fail to execute unless the
        --applyChangeTo argument is provided with the value server-group-force. Only applies if the batch
        file argument is also supplied.
    --dry-run
        Validate configuration changes but do not apply them. This option can only be used along with the
        -F/--batch-file option
    --propertiesFilePath {propertiesFilePath}
        Path to the file that contains default property values used for command-line arguments
    --noPropertiesFile
        Specify that no properties file will be used to get default command-line argument values
    --script-friendly
        Use script-friendly mode

  General Options

    -V, --version
        Display Directory Server version information
    -?, -H, --help
        Display general usage information
    --help-ldap
        Display help for using LDAP options
    --help-sasl
        Display help for using SASL options
    --help-debug
        Display help for using debug options

Examples

  Start dsconfig in interactive mode:

    dsconfig

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

    dsconfig --no-prompt --bindDN uid=admin,dc=example,dc=com \
         --bindPassword password 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

  Use batch mode to read and execute a series of commands in a batch file:

    dsconfig --bindDN uid=admin,dc=example,dc=com --bindPassword password \
         --no-prompt --batch-file /path/to/config-batch.txt

  List information about all available configuration properties for all objects, including inherited
properties:
```

```
    dsconfig list-properties --offline --inherited

  For examples and help with LDAP options see --help-ldap.  For help with SASL authentication, see --help-
sasl
```

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.

   Result:

   You are prompted for connection and authentication information to the PingDirectoryProxy 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 to present 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 Installing the PingDirectory server 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.

   Choose from:

   - 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 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 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.**

   Example:

   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 PingDirectory 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.

dsconfig non-interactive mode command">

Getting the equivalent `dsconfig` non-interactive mode command

Find the command you want to use in non-interactive command-line mode by looking it up in interactive mode.

Steps

1. Use `dsconfig` in interactive mode to make changes to a configuration, but do not enter the letter `f` to apply the changes.
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 message.

```
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 the command is presumed to be used to connect to a different remote host.

Using dsconfig batch mode

Configure the server in `dsconfig` batch mode.

About this task

The PingDirectory server provides a `dsconfig` batching mechanism that reads multiple `dsconfig` invocations from a file and executes them sequentially.

The batch file provides advantages over standard scripting by minimizing LDAP connections and 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 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 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 server also provides a `docs/sun-ds-compatibility.dsconfig` file for migrations from Oracle to PingDirectory 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 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.

Example:

```
# 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 the PingDirectory server or the 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 the PingDirectory server or the 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.

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 primary and secondary architecture for mirroring shared data across the topology.

All writes and updates are forwarded to the primary, 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 primary requirements and selection

A topology primary 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 primary sends a digest of its subtree contents. If the node's digest differs from the primary's, the server node knows it is not synchronized. The servers pull the entire subtree from the primary if they detect that they are not synchronized.

A server detects it is not synchronized with the primary under the following conditions:

- The server's subtree digest differs from the primary's digest at the end of its periodic polling interval.
- One or more servers have been added to or removed from the topology.

The primary 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 primary, the topology data is reviewed from all available servers, every five seconds by default, to determine if any new information identifies a better server primary. If a new server can be the primary and no other servers have indicated that they should be the primary, it will communicate its eligibility to the other servers. This ensures that all servers accept the same primary at approximately the same time, within a few milliseconds of each other. If there is no better primary, the initial primary maintains the role.

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

- A majority of servers is reachable from that primary.

  The primary server itself is considered while determining this majority.

- There is only a single primary in the entire topology.

If either of these conditions is not met, the topology is without a primary and the peer polling frequency is reduced to 100 milliseconds to find a new primary as quickly as possible. If there is no primary 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 primary 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 primaries, 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 primary because most of the servers are not available. A server with this setting enabled is assigned the role of primary if no suitable primary 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 primary 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 primary is found, the polling frequency adjusts to 100 milliseconds until a new primary 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 primary on all of the servers in the topology. The default is 10 seconds, but updates can take up to twice as long if primary 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 primary, if the primary is not known

- The instance name of the current primary or if a primary is not set, a description stating why a primary is not set
- A flag indicating which server thinks that it should be the primary
- A flag indicating which server is the current primary
- A flag indicating a server that was forced as primary
- 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 primary or another node in the topology if the primary is unknown
- The amount of time in milliseconds that multiple primaries were detected by this server
- The amount of time in milliseconds that no suitable server is found to act as primary
- 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 primary:

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

Using the Configuration API

The 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.

Result:

The API is made available on the HTTPS Connection Handler's host:port in the `/config` context. Because of 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	Comparabledsconfig 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.backgroundReadLimit=0" }`	`$ dsconfig set-backend-prop --backend-name userRoot \ --add je-property:je.env.backgroundReadLimit=0`
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\"]" }`	`$ dsconfig set-backend-prop --backend-name userRoot \ --remove je-property:je.cleaner.adjustUtilization=false`

Operation	PATCH request	Comparabledsconfig modify-[object]options
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" : "id2childrenIndexEntryLimit" }`	`$ dsconfig set-backend-prop --backend-name userRoot \ --reset id2childrenIndexEntryLimit`

Example

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"
  } ]
}
```

Example

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`.

Example

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.

Example

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.

Example

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.

Example

```
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 because of 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.

Example

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."
}
```

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 PingDirectoryProxy server instances into the server group labeled "group-one".

Steps

1. Create a group called "group-one" using `dsconfig`.

   Example:

   ```
   $ 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.

   Example:

   ```
   $ 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.

   Example:

   ```
   $ 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.

   Example:

   ```
   $ 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.

   Example:

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

Result:

```
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`.

   Example:

   ```
   $ 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.

   Example:

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

   Result:

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

9. View the property on the second PingDirectory server instance.

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

   Example:

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

   Result:

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

Generating a summary of configuration components

The `config-diff` tool generates 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 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`

  Example:

  `$ 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`.

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 caused by unnecessary DNS lookups, follow these recommendations:
    1. Maintain a connection pool in the client app rather than opening new connections for each bind.
    2. Add appropriate records to DNS, including PTR records.
    3. Add `options timeout:1` or `options single-request` in the `/etc/resolv.conf` file.
    4. If IPv6 requests are causing issues, add `-Djava.net.preferIPv4Stack=true` to the `start-server.java-args` line in 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 directory 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 for the PingDirectory server and the PingDirectoryProxy server
- `ds-auth-allowed-address`: Restricting access through operational attributes in user entities for the PingDirectory server
- Connection Criteria: Restricting server access based on client IP address for the PingDirectory server and the PingDirectoryProxy server
- Connection Handlers: Restricting server access using the connection handlers for a configuration reference guide for all servers

dsconfig">

Configuring traffic through a load balancer using `dsconfig`

About this task

To configure the directory server to get traffic through a load balancer and to record the actual client's IP address:

Steps

1. Edit the HTTP or HTTPS connection handler object and set `use-forwarded-headers` to `true` by running `dsconfig`.

   Example:

   ```
   dsconfig set-connection-handler-prop \
       --handler-name "HTTPS Connection Handler" \
       --set use-forwarded-headers:true
   ```

2. To finalize the changes to the HTTP or HTTPS connection handler, use `dsconfig` to restart the connection handler.

   Example:

   ```
   dsconfig set-connection-handler-prop \
       --handler-name "HTTPS Connection Handler" \
       --set enabled:false

   dsconfig set-connection-handler-prop \
       --handler-name "HTTPS Connection Handler" \
       --set enabled:true
   ```

3. To provide the `X-Forwarded-*` information to your load balancer, consult your provider's guide on configuring the load balancer settings.

Managing root user accounts

The directory 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.

  Example:

  ```
  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 the servers, 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 PingDirectory server contains a privilege subsystem that allows for a more fine-grained control of privilege assignments.

> **Note**
>
> Creating restricted root user accounts requires assigning privileges and necessary access controls for actions on specific data or backends. Access controls are determined by how the directory is configured and the structure of your data. See Managing access control 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.
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.

Privilege	Description
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 PingDirectory 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 can 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

The PingDirectoryProxy server defines locations, both for the LDAP external servers and the directory proxy 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 admin console or the command line.

The PingDirectoryProxy server itself is also associated with a location. This location is specified in the global configuration properties of the server. If the load balancing algorithm's `use-location` property is set to true, then the load balancing component of the PingDirectoryProxy server refers to the server's location to determine the external servers it prefers to communicate with.

Modifying locations using dsconfig

Steps

1. To configure existing LDAP external server locations, use the `dsconfig` tool.

```
$ bin/dsconfig
```

2. Enter the host name or IP address for your PingDirectoryProxy server, or press Enter to accept the default, `localhost`.

```
Directory Proxy Server host name or IP address [localhost]:
```

3. Enter the number corresponding to how you want to connect to the PingDirectoryProxy 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. Enter the port number for your PingDirectoryProxy server, or press Enter to accept the default, `389`.

```
Directory Proxy Server port number [389]:
```

5. Enter the administrator's bind distinguished name (DN) or press Enter to accept the default `cn=Directory Manager`, and then enter the password.

```
Administrator user bind DN [cn=Directory Manager]:
Password for user 'cn=Directory Manager':
```

6. In the `PingDirectoryProxy 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 PingDirectoryProxy server instance, in this example the East location.

   Example:

Operations involving communications with other servers might 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. To finish the operation, enter `f`.

```
Enter choice [b]: f
```

Configuring locations using `dsconfig`

Steps

1. To configure the LDAP external server locations, use the `dsconfig` tool.

```
$ bin/dsconfig
```

2. Enter the host name or IP address for your PingDirectoryProxy server, or press Enter to accept the default, `localhost`.

```
Directory Proxy Server host name or IP address [localhost]:
```

3. Enter the number corresponding to how you want to connect to the PingDirectoryProxy 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. Enter the port number for your PingDirectoryProxy server, or press Enter to accept the default, `389`.

```
Directory Proxy Server port number [389]:
```

5. Enter the administrator's bind distinguished name (DN) or press Enter to accept the default, `cn=Directory Manager`, and then enter the password.

```
Administrator user bind DN [cn=Directory Manager]:
Password for user 'cn=Directory Manager':
```

6. In the `PingDirectoryProxy 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.

   Example:

   This example demonstrates configuring a location called `East`.

```
>>>>  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. To finish configuring the location, enter `f`.

9. Edit the configuration of an existing location, in this example a location named `East`.

   Example:

```
>>>> 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
```

10. Define the `preferred-failover-location` property for `East`.

    This property provides alternate locations that can be used if servers in this location aren't available. If more than one location is provided, the PingDirectoryProxy server tries the locations in the order listed.

    Example:

```
 >>>> 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
```

11. Verify and apply your change to the property.

    Example:

```
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
```

Configuring batched transactions

Configure the PingDirectoryProxy server to use batched transactions in both simple and entry-balanced configurations.

About this task

The batched transactions feature supports two implementations: the standard LDAP transactions per RFC 5805 and the PingDirectoryProxy proprietary implementation, known as the multi-update extended operation.

Batched transactions can be used through the PingDirectoryProxy server in both simple and entry-balanced configurations, but only in cases where all operations within the transaction request can be processed within the same backend server and within the same Berkeley DB JE backend.

> **Note**
>
> Batched transactions can't be processed across multiple servers or multiple PingDirectory server backends.

You can submit multiple updates in a single request. These updates can be processed either as individual operations or as a single batch. When the PingDirectoryProxy server receives a Start Batched Transaction request, it queues 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 PingDirectory server.

You can include `add`, `delete`, `modify`, `modify DN`, and `password modify` extended operations to 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 processing or to cancel any remaining operations. If the operations aren't canceled, you can configure the server to process all operations as a single unit.

Because of this use of multi-update, you must configure the external PingDirectory server to allow multi-update extended requests made by the PingDirectoryProxy server on behalf of the DN submitting the batched transaction.

For example, the following PingDirectory server `dsconfig` command grants anonymous access to the multi-update extended request.

```
$ 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";)'
```

> **Note**
>
> The submitter of the request still needs access rights for the individual operations within the multiple-update.

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`, or `password modify` operations as a single atomic unit.

Steps

1. Configure batched transactions using the `dsconfig` command.

   Example:

   ```
   $ 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 PingDirectoryProxy 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, see About LDAP health checks.

About the default health checks

By default, the PingDirectoryProxy server has two health check instances enabled for use on all servers:

- Consume Admin Alerts. This health check detects administrative alerts from the PingDirectory server, as soon as they are issued, by maintaining an LDAP persistent search for changes within the `cn=alerts` branch of the PingDirectory server. When the PingDirectoryProxy server is notified by the PingDirectory server of a new alert, it immediately retrieves the base `cn=monitor` entry of the PingDirectory server. If this entry has a value for the `unavailable-alert-type` attribute, then the PingDirectoryProxy server will consider it unavailable. If this entry has a value for the `degraded-alert-type` attribute, then the PingDirectoryProxy server will consider it degraded.
- Get Root DSE. This health check detects if the root DSE entry exists on the LDAP external server. Since this entry always exists on a PingDirectory server, the absence of the entry suggests that the LDAP external server could 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 PingDirectoryProxy 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 on the response time in which the expected results were returned. 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 can monitor the percentage of time that worker threads in backend servers devote to processing requests.

dsconfig">

Configuring a health check using `dsconfig`

Steps

1. To configure the LDAP external server locations, use the `dsconfig` tool.

```
$ bin/dsconfig
```

2. Enter the host name or IP address for your PingDirectoryProxy server, or press Enter to accept the default, `localhost`.

```
Directory Proxy Server host name or IP address [localhost]:
```

3. Enter the number corresponding how you want to connect to the PingDirectoryProxy 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. Enter the port number for your PingDirectoryProxy server, or press Enter to accept the default, `389`.

```
Directory Proxy Server port number [389]:
```

5. Enter the administrator's bind distinguished name (DN) or press Enter to accept the default `cn=Directory Manager`. Then enter the password.

```
Administrator user bind DN [cn=Directory Manager]:
Password for user 'cn=Directory Manager':
```

6. In the `PingDirectoryProxy main menu`, enter the number corresponding to LDAP health checks.
7. Enter the number to create a new LDAP Health Check, and then press `n` to create a new health check from scratch.
8. Select the type of health check you want to create.

   Example:

   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
```

9. Enter a name for the new health check.

   Example:

   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
```

10. To enable the new health check, enter `1`.

```
>>>> 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
```

11. Configure the properties of the health check.

    You might need to modify the `base-dn` property as well as one or more response time thresholds for non-local external servers, accommodating WAN latency.

    Example:

    The following example 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 PingDirectoryProxy server to communicate with the server properly.

PingDirectoryProxy includes a tool, `prepare-external-server`, for configuring communication between the PingDirectoryProxy 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 doesn't make configuration changes to the local PingDirectoryProxy 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 PingDirectoryProxy server during configuration, `cn=Proxy User` by default.

Important

You shouldn't use `cn=Directory Manager` as the account to use for communication between the PingDirectoryProxy server and the PingDirectory server. For security reasons, the account used to communicate between the PingDirectoryProxy server and the PingDirectory server shouldn't be directly accessible by clients accessing the PingDirectoryProxy server. The account that you choose should meet the following criteria:

- For all server types, it shouldn't exist in the PingDirectoryProxy server but only in the backend directory server instances.
- For the PingDirectory server, this user should be a root user.
- For the PingDirectory server, this user shouldn't 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 don't 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 PingDirectoryProxy server's trust store, you must include the `--proxyTrustStorePath` option and either the `--proxyTrustStorePassword` or the `--proxyTrustStorePasswordFile` option.

The default location of the PingDirectoryProxy server's trust store is `config/truststore`. The pin is encoded in the `config/truststore.pin` file.

The following example prepares a PingDirectory server on the remote host `ds-east-01.example.com`, listening on port 1389 for access by the PingDirectoryProxy server using the default user account `cn=Proxy User`, as shown in the following example.

```
prepare-external-server --hostname ds-east-01.example.com \
--port 1389 --baseDN dc=example,dc=com --proxyBindPassword secret
```

When the `prepare-external-server` command is executed, it creates the `cn=Proxy User` Root distinguished name (DN) entry as well as an access control rule in the PingDirectory server to grant the proxy user the proxy access right.

Important

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 access control instruction (ACI) entries for these servers.

Configuring server communication using the `prepare-external-server` tool

The following example illustrates how to run the `prepare-external-server` tool to prepare a PingDirectory server.

About this task

In this example, the server is configured on the remote host `ds-east-01.example.com`, listening on port 1636.

The PingDirectory server is being accessed by a PingDirectory server that uses the default user account `cn=Proxy User,cn=Root DNs,cn=config`. Because a password to the trust store isn't provided, the trust store defined in the `--proxyTrustStorePath` is referenced in a read-only manner.

Steps

- To prepare the PingDirectory server, use the `prepare-external-server` tool.

  Example:

  ```
  $ ./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
  ```

- Follow the prompts to set up the external server.

```
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's 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`

Use the `dsconfig` tool to create and configure external servers. Then, specify the host name, connection method, port number, and bind distinguished name (DN), as described in previous procedures.

Steps

1. Run the `dsconfig` command.

   ```
   $ bin/dsconfig
   ```

2. In the `PingDirectoryProxy main menu`, enter the number corresponding to external servers.
3. Enter the number to create a new external server.
4. Select the type of server you want to create.

   Example:

   This example creates a new Ping Identity PingDirectory 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
```

5. Specify a name for the new external server.

   Example:

   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
```

6. Configure the host name or IP address of the target LDAP external server.

   Example:

```
Enter a value for the 'server-host-name' property:east1.example.com
```

7. Configure the location property of the new external server.

   Example:

```
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
```

8. Define the bind DN and bind password.

   Example:

```
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:
```

9. To finish the operation, enter `f`.

```
Enter choice [b]: f
```

Result

```
The Ping Identity DS external server was created successfully.
```

Next steps

After adding the server, run the `prepare-external-server` tool to configure communications between the PingDirectoryProxy server and the Ping Identity PingDirectory server.

For more information, see Preparing two new external servers using the prepare-external-server tool.

Configuring authentication with a SASL external certificate

By default, the PingDirectoryProxy server authenticates to the PingDirectory server using LDAP simple authentication with a bind DN and a password. You can configure the PingDirectoryProxy server to use Simple Authentication and Security Layer (SASL) EXTERNAL to authenticate to the PingDirectory server with a client certificate.

Before you begin

Install and configure the PingDirectoryProxy server instances to communicate with the backend PingDirectory server instances using either SSL or StartTLS.

Steps

1. Create a Java KeyStore (JKS) that includes a public and private key pair for a certificate that the PingDirectoryProxy server instances will use to authenticate to the PingDirectory instances.
    1. Run the following command in the instance root of one of the PingDirectoryProxy server instances.

```
$ 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. When prompted for a key store password, enter a strong password to protect the certificate.
    3. When prompted for the key password, press Enter to use the key store password to protect the private key.
2. Use a text editor to create a `config/proxy-user-keystore.pin` file containing a single line that is the key store password provided in the previous step.
3. If there are other PingDirectoryProxy server instances in the topology, copy the `proxy-user-keystore` and `proxy-user-keystore.pin` files into the `config` directory for all instances.
4. To export the public component of the proxy user certificate to a text file, run the following command.

```
$ 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.
    1. 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 proxy-user-cert \
  -file config/proxy-user-cert.txt
```

    2. When prompted for the keystore password, enter the password contained in the `config/truststore.pin` file.
    3. When prompted to trust the certificate, enter `yes`.

6. To update the configuration for each PingDirectoryProxy 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. To update the configuration for each LDAP external server in each PingDirectoryProxy server instance to use the newly-created key manager provider and 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"
```

   Result:

   After these changes, the PingDirectoryProxy server reestablishes connections to the LDAP external server and authenticates with SASL EXTERNAL.

8. Verify that the PingDirectoryProxy server can communicate with all backend servers by running the `bin/status` command.

   Result:

   All of the servers listed in the "--- LDAP External Servers ---" section are available.

9. Review the PingDirectory server access log.

   The BIND RESULT log messages used to authenticate the connections from the PingDirectoryProxy server include the following:

   - `authType="SASL"`
   - `saslMechanism="EXTERNAL"`
   - `resultCode=0`
   - `authDN="cn=Proxy User,cn=Root DNs,cn=config"`

Configuring load balancing

You can distribute the load on your PingDirectoryProxy server using one of the load-balancing algorithms provided with the PingDirectoryProxy server. By default, the PingDirectoryProxy 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 PingDirectoryProxy server prefers available servers over degraded servers. This means that if at all possible, the PingDirectoryProxy 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 PingDirectoryProxy 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 PingDirectoryProxy server to prefer local DEGRADED health servers to servers in a failover location. See the online PingDirectoryProxy Configuration Reference Guide for more information on the `prefer-degraded-servers-over-failover` property.

The PingDirectoryProxy 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 might 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 distinguished names (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 are consistently routed to the same server, but requests targeting a different portion of the data might be sent to a different server.

Load spreading is useful for deployments:

- Where the directory information tree (DIT) contains a large number of branches below a common parent
- Where 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, load spreading can 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 or values of this property are the base DN or DNs below which the tenant entries reside.

In a deployment with a DIT as in the following example, 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 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 is handled using the default behavior.

When the `load-spreading-base-dn` property is configured with one or more values, the load-balancing algorithm continues to generate the same list of lists, but the order of the servers within each list is 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 Relative Distinguished Name (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, the order of servers in each of those lists is 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 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 yields an integer value that is between 0 and `(list.size() - 1)`, inclusive.

   > **Note**
   >
   > `(list.size() - 1)` is the number of servers minus 1.

6. If the modulus computed is equal to 0, no further action is necessary. If not, move several 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 an order of ds3, ds1, ds2.

While this algorithm spreads 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 still prioritizes based on location and health check state, so the load is generally spread only across the available servers in the same location as the PingDirectoryProxy server. 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 are targeted more heavily than others because they have more entries or because their entries are accessed more frequently. The modulo operation might not result in an even distribution across those servers.

dsconfig">

Configuring load balancing using `dsconfig`

Create and configure a load-balancing algorithm with `dsconfig.`

About this task

To create and configure a load-balance algorithm:

Steps

1. Run the `dsconfig` tool.

   When prompted, specify the host name, connection method, port number, and bind distinguished name (DN).

   Example:

   ```
   $ bin/dsconfig
   ```

2. In the PingDirectoryProxy main menu, enter the selection for `load-balancing algorithms`.
3. To create the Load Balancing Algorithm, in the `Load Balancing Algorithm menu`, select an option.

   Choose from:

   - To use an existing load-balancing algorithm as a template, enter `t`.
   - To create a new load-balancing algorithm from scratch, enter `n`.

     Example:

     In this example, a new Load Balancing Algorithm is created 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. In `Select the type of Load Balancing Algorithm menu`, select the type of load-balancing algorithm that you want to create.

   Depending on the type of algorithm you select, you're guided through a series of configuration properties, such as providing a name and selecting an LDAP external server.

   Example:

   ```
   >>>> 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 and enter `f`.

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 and 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 isn't a match, then the default load-balancing algorithm is used.

Preferring failover LBA for write operations

You can configure the PingDirectoryProxy 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 PingDirectoryProxy server deployments, you should use a failover load-balancing algorithm for at least `add`, `delete`, and `modifyDN` operations

Each proxying request processor configured in the PingDirectoryProxy server uses a load-balancing algorithm to choose which PingDirectoryProxy 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 PingDirectoryProxy server.
- If the PingDirectoryProxy server is degraded for any reason, such as having a local database (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
- 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 prefer directory servers in the same location as the PingDirectoryProxy server. However, they differ in the criteria they use to choose between available servers. The following sections describe the algorithms in more detail.

Choosing between available servers

The failover load-balancing algorithm sends all operations to a single server until it's unavailable and then sends all operations to the next preferred server, and so forth. This algorithm provides the most consistent view of the topology to clients because all clients, at least those in the same location as the PingDirectoryProxy server, see the same, up-to-date view of the data, but it leaves unused capacity in the failover instances because most topologies include multiple directory server replicas within each data center.

The fewest operations load-balancing algorithm efficiently distributes traffic among multiple servers by choosing to send each operation to the server that has the fewest number of outstanding operations, which is the server from the PingDirectoryProxy 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 because 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 and limits 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, 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, you should configure the PingDirectoryProxy server to route dependent requests to the same server.

The server affinity feature

The server affinity feature achieves this in some environments but not in all because the affinity is tracked independently by each PingDirectoryProxy server instance, and some clients send requests to multiple proxies. It's common for a client not to connect to the PingDirectoryProxy servers directly but instead to connect through a network load balancer, which in turn opens connections to the PingDirectoryProxy servers. Each individual client connection is established to a single PingDirectoryProxy server so that operations on that connection are routed to the same PingDirectoryProxy server, and server affinity configured within the PingDirectoryProxy server ensures those operations are routed to the same directory servers.

However, many clients establish a pool of connections that are reused across operations, and within this pool, connections are established through the load balancer to different PingDirectoryProxy servers. Dependent operations sent on different connections could then be routed to different PingDirectoryProxy servers and then on to different PingDirectoryProxy servers.

> **Note**
>
> For more information about the server affinity feature, see Configuring Server Affinity.

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 (`add` , `delete` , `modify` , and `modifyDN`) could be routed using a failover load-balancing algorithm while all other operations (`bind` , `search` , and `compare`) use a fewest operations load-balancing algorithm.

If there are clients that are particularly sensitive to reading entries immediately after modifying them, additional connection criteria can be specified for all operations from those clients using the failover load-balancing algorithm.

> **Note**
>
> Routing all write requests to a single server in a location instead of evenly across servers doesn't limit the overall throughput of the system because all servers ultimately have to process all write operations either from the client directly or through replication.

Replication conflicts

Another benefit of using the failover load-balancing algorithm for write operations is reducing replication conflicts. The PingDirectory server follows the traditional LDAP replication model of eventual consistency. This provides 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 `modifyDN` operations can't always be resolved automatically and might require manual involvement from an administrator. By routing all write operations (or at least `add` , `delete` , and `modifyDN` operations) to a single server, replication conflicts can be avoided.

> **Note**
>
> When using a failover load-balancing algorithm, consider the following:
>
> - When using the failover load-balancing algorithm in a configuration with multiple locations, the load-balancing algorithm fails 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 PingDirectoryProxy server's configuration is used to decide which remote location to fail over to. It's 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 PingDirectoryProxy 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 PingDirectoryProxy 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 PingDirectoryProxy 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. This example can be extended to support alternate criteria and more complex topologies using multiple locations or entry balancing.

> **Note**
>
> This example uses a basic deployment of a PingDirectoryProxy server fronting three directory servers: `ds1.example.com`, `ds2.example.com`, and `ds3.example.com`.

Steps

1. To create a location, run `dsconfig` with the `create-location` option.

   Example:

   ```
   dsconfig create-location --location-name Austin
   ```

2. To update the failover location for your server, run `dsconfig` with the `set-location-prop` option.

   Example:

   ```
   dsconfig set-location-prop --location-name Austin
   ```

3. To set the location as a global configuration property, run `dsconfig` with the `set-global-configuration-prop` option.

   Example:

   ```
   dsconfig set-global-configuration-prop --set location:Austin
   ```

4. To set up the health checks for each external server, run `dsconfig` with the `create-ldap-health-check` option.

   Example:

   ```
   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. To create the external servers, run `dsconfig` with the `create-external-server` option.

Example:

```
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. To create a load-balancing algorithm, run `dsconfig` with the `create-load-balancing-algorithm` option.

   Example:

   This example creates 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. To create a request processor, run `dsconfig` with the `create-request-processor` option.

   Example:

   This example creates 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. To create a subtree view, run `dsconfig` with the `create-subtree-view` option.

   Example:

   This example creates 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. To update the client connection policy, run `dsconfig` with the `set-client-connection-policy-prop` option.

   Example:

   This example updates 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. To create a new request criteria object to match all write operations, run `dsconfig` with the `create-request-criteria` option.

    Example:

    ```
    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. To create a new failover load-balancing algorithm listing the servers that should be included, run `dsconfig` with the `create-load-balancing-algorithm` option.

    Example:

    In this example, the order in which the servers are listed 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. To tie the request criteria and the failover load-balancing algorithm together into a criteria-based load-balancing algorithm, run `dsconfig` with the `create-criteria-based-load-balancing-algorithm` option.

    Example:

    ```
    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. To update the proxying request processor to use the criteria-based load-balancing algorithm, run `dsconfig` with the `set-request-processor-prop` option.

    Example:

    ```
    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
    ```

Result

The PingDirectoryProxy server routes all write operations to `ds1.example.com` as long as it's available and then to `ds2.example.com` if it's not. The PingDirectoryProxy server routes other types of operations, such as searches and binds, to all three servers using the fewest operations load-balancing algorithm.

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 task but configure connection criteria instead of request criteria. These connection criteria identify clients that could be impacted by replication latency. These clients 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 requeries the directory for an updated list of users, the just-deleted entry must not be included.

To configure a criteria-based load balancing algorithm:

Steps

1. Create the new failover load-balancing algorithm using `dsconfig` with the `create-load-balancing-algorithm` option.

   > **Note**
   >
   > The failover load-balancing algorithm should list the same set of servers as the existing fewest operation load-balancing algorithm.

   Example:

   ```
   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` with the `create-connection-criteria` option.

   Example:

```
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 using `dsconfig` with the `create-criteria-based-load-balancing-algorithm` option.

   Use the load balancing algorithm and connection criteria created in steps 1-2.

   Example:

   ```
   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` with the `set-request-processor-prop` option.

   Example:

   ```
   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

After 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 that is 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.

> **Note**
>
> If there is an affinity to a particular server and that server is classified as available, that server is always the first in the list.

Next, the PingDirectoryProxy 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
- Location, with backend servers in the same location as the PingDirectoryProxy server most preferred, then servers in the first failover location, then the second, and so forth

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 or availability over location, such as specifying if a local degraded server is more or less preferred than a remote available server.

By default, the algorithm prefers 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 PingDirectoryProxy server does both proactive and reactive health checking.

Proactive health checking

The PingDirectoryProxy server periodically (by default, every 30 seconds) runs a full set of tests against each backend server. The result of these tests are used to determine the overall health check state (available, degraded, or unavailable) and score (an integer value from 10 to 0).

Reactive health checking

The PingDirectoryProxy server kicks 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. 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 subsequent proactive health check determines that it has recovered. If a server is determined to be degraded, it might 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 are considered for all subsequent requests routed to the load-balancing algorithm. If a server is considered degraded, then it's immediately 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's not 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. 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. The server's `start-server` tool outputs 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 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 optional HTTP log publisher.

You must configure the HTTP servlet extension and log publisher before configuring the HTTP connection handler. The log publisher is optional but in most cases, you should configure one or more logs to troubleshoot any issues with your HTTP connection.

Steps

1. To configure your HTTP servlet extensions, run `dsconfig` with the `create-http-servlet-extension` option.

   Example:

   This 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. To configure an HTTP log publisher, run `dsconfig` with the `create-log-publisher` option.

   Example:

   This example configures one log publisher 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, run `dsconfig` with the `create-connection-handler` option to specify the HTTP servlet extension and log publishers.

> **Note**
>
> Some configuration properties can be updated while the HTTP connection handler is enabled while others, such as `listen-port`, require that the HTTP connection handler be disabled and then re-enabled for the change to take effect.

Example:

```
$ 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`.

Example:

```
$ 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 PingDirectory 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`.

  Example:

  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
  ```

  Example:

  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.

Example:

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.

Example:

```
$ 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 that also have LDAP-specific log messages. The response to the API call includes a `Correlation-Id` header with the value `ee919049-6710-4594-9c66-28b4ada4b127`.

```
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"
     ]
 }
```

The correlation ID can be used to search the HTTP trace log for matching log records, as in the following example.

```
$  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="{pingdir}  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="{pingdir}
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="{pingdir}
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, as in the following example.

```
$  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 the PingDirectoryProxy server to use an HTTP proxy server

Some organizations configure their network so that internal systems can't directly access the Internet but instead must use an HTTP proxy server to access external services.

The PingDirectoryProxy server now supports using an HTTP proxy server in conjunction with the following components:

- The Amazon Key Management Service cipher stream provider
- The Amazon Secrets Manager cipher stream provider
- The Amazon Secrets Manager passphrase provider
- The Amazon Secrets Manager password storage scheme
- The Azure Key Vault cipher stream provider
- The Azure Key Vault passphrase provider
- The Azure Key Vault password storage scheme
- The PingOne pass-through authentication plugin
- The Pwned Passwords password validator
- The Twilio alert handler

- The Twilio OTP delivery mechanism
- The UNBOUNDID-YUBIKEY-OTP SASL mechanism handler

Setting up the server to use an HTTP proxy server involves two steps, which are described in the following sections.

1. Creating an HTTP proxy external server in the configuration.
2. Configuring the appropriate component(s) to use the HTTP proxy server.

Creating an HTTP proxy external server

An HTTP proxy external server configuration object provides information about a proxy server that should be used. At present, we only support HTTP proxy servers (which can handle both unencrypted HTTP and encrypted HTTPS connections), with or without authentication.

HTTP proxy external server definitions support the following configuration properties:

`server-host-name`

The resolvable name or IP address of the HTTP proxy server to use. This is required.

`server-port`

The port on which the HTTP proxy server is listening for connections. This is required.

`basic-authentication-username`

The username to use if the proxy server requires authentication. This should be omitted if the proxy server doesn't require authentication.

`basic-authentication-passphrase-provider`

The passphrase provider to use to obtain the password to use if the proxy server requires authentication. This should be omitted if the proxy server doesn't require authentication.

For example, you can use a configuration change like the following to create an HTTP proxy external server that doesn't require authentication:

```
dsconfig create-external-server \
     --server-name "Example HTTP Proxy Server" \
     --type http-proxy \
     --set server-host-name:proxy.example.com \
     --set server-port:3128
```

Configuring server components to use the HTTP proxy external server

Merely defining an HTTP proxy external server in the configuration doesn't cause the server to use that proxy server for anything. Instead, you must indicate which components should use that proxy server.

This is necessary because you might only need to use an HTTP proxy server for certain components (for example, it might be necessary when accessing external web services, but not for services on the internal private network).

All of the components that support the use of an HTTP proxy server offer an `http-proxy-external-server` configuration property whose value should be the name of the appropriate HTTP proxy external server definition in the configuration. For example, to update the Pwned Passwords password validator to use the HTTP proxy server defined in the "Example HTTP Proxy Server" configuration object, use a configuration change like the following:

```
dsconfig set-password-validator-prop \
     --validator-name "Pwned Passwords" \
     --set "http-proxy-external-server:Example HTTP Proxy Server"
```

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 PingDirectoryProxy server provides the following types of data transformations:

- Attribute mapping. The attribute mapping transformation rewrites client requests so that references to one attribute type can be replaced with an alternate attribute type. The PingDirectoryProxy server can perform extensive replacements, including attribute names used in DNs and attribute names encoded in the values of several 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.

  > **Important**
  >
  > Before adding attribute mapping transformations, you must define both the source and target attribute types in the local schema. This requirement ensures that the server handles the attribute values correctly.
  > If any existing attribute mapping transformations reference attributes that don't meet this requirement, the server logs a warning on startup. You should review the logs and properly define any such attributes in the local schema.

- Default value. The default value transformation instructs the PingDirectoryProxy 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. The DN mapping 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. The Groovy scripted 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 PingDirectoryProxy server.

- Suppress attribute. The suppress attribute 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. The suppress entry 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. The simple-to-external bind proxy transformation can 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 might or might not be processed, depending on how you configure the server.
- Third-party scripted. The third-party scripted 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 PingDirectoryProxy server.

dsconfig">

Configuring proxy transformations using `dsconfig`

Create and configure a proxy data transformation.

About this task

You can use PingDirectoryProxy to perform one of the following transformations:

- Attribute mapping
- Default value
- DN mapping
- Groovy scripted
- Suppress attribute
- Suppress entry
- Simple-to-external bind
- Third-party scripted

Steps

To create and configure a proxy transformation:

1. Run the `dsconfig` tool.

   Example:

   ```
   $ bin/dsconfig
   ```

2. When prompted, enter the connection parameters to the server.

   Connection parameters can include:

   - Host name

- Connection method
- Port
- Bind distinguished name (DN)
- Bind DN password

3. In the PingDirectoryProxy server main menu, enter the selection for `Proxy transformations`.
4. In the `Proxy Transformation menu`, enter the selection for `Create a new proxy transformation`.
5. Select the type of proxy transformation you want to create. Enter a name for the new transformation.

   Example:

   This example creates an attribute mapping transformation.

   ```
    >>>> Enter a name for the Attribute Mapping Proxy Transformation that you
   want to create: userid-to-uid
   ```

6. To choose if the proxy transformation is enabled by default or not, select `true` or `false`.

   Example:

   ```
   Select a value for the 'enabled' property:

     1) true
     2) false

     ?) help
     c) cancel
     q) quit

   Enter choice [c]: 1
   ```

7. Enter the name of the client attribute that you want to remap to a target attribute.

   > **Important**
   >
   > This attribute must not be equal to the target attribute. Both the source and target attribute types must be defined in the local schema.

   Example:

   ```
   Enter a value for the 'source-attribute' property: userid
   ```

8. Enter the name of the target attribute to which the client attribute should be mapped.

   Example:

   ```
   Enter a value for the 'target-attribute' property: uid
   ```

   Result:

The properties of your new proxy transformation are displayed.

9. (Optional) To make any modifications, enter the selection for the desired property.
10. To finish the creation of the proxy transformation, enter `f`.

    Example:

    ```
    Enter choice [b]: f
    ```

Next steps

The transformation must be assigned to a request processor. For information on creating and configuring request processors, see Configuring request processors.

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 PingDirectoryProxy 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 can be configured to provide high availability and load balancing, and various transformations can 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.

dsconfig">

Configuring request processors using `dsconfig`

About this task

To create and configure a request processor:

Steps

1. Run the `dsconfig` tool.

    Example:

    ```
    $ bin/dsconfig
    ```

2. When prompted, enter the connection parameters to the server.

Connection parameters can include:

- Host name
- Connection method
- Port
- Bind distinguished name (DN)
- Bind DN password

3. In the PingDirectoryProxy server's main menu, enter the selection for `Request Processor configuration`.
4. Enter the selection for `Create a new request processor`.
5. Select an existing request processor to use as a template for creating a new one or enter `n` to create one from scratch.
6. Select an existing load balancing algorithm or create a new one.

Result:

The following example shows the configuration of a new proxying request processor after the 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
```

7. Review the configuration properties of the new request processor. To finish, enter `f`.

> **Note**
>
> To use the request processor, you must associate it 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 PingDirectoryProxy to forward these controls correctly.

Steps

- Configure the `supported-control-oid` property to define the request OID of the LDAP control.

  Result:

  The PingDirectoryProxy server updates the root DSA-specific entry (DSE) `supportedControl` attribute with the values entered for the `supported-control-oid` property.

Configuring server affinity

About this task

The PingDirectoryProxy 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 PingDirectoryProxy server:

- Client Connection. Requests from the same PingDirectoryProxy server client connection are consistently routed to the same external server.
- Client IP. PingDirectoryProxy 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 might 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.

   Example:

```
$ bin/dsconfig
```

2. In the PingDirectoryProxy 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.

   Example:

```
>>>> 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 can be set and the types of operations for which affinity can 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.

   Example:

```
>>>> >>>> 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. After it has been 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 admin 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 PingDirectoryProxy server's 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.

   Example:

   ```
   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.

   Example:

   ```
   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.

   Example:

```
>>>> 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
```

Result:

After they have been 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.

`required-operation-request-criteria`

This property causes the server to reject any requests that do not match the referenced request criteria.

`prohibited-operation-request-criteria`

This 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. A client connection policy can't 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, it is terminated.

> **Note**
>
> The PingDirectoryProxy 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 distinguished name (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 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 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
- How the policy is evaluated
- Configuring a client connection policy using the console
- Configuring a client connection policy using dsconfig
- Restricting server access based on client IP address

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 several 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.

dsconfig">

Configuring a client connection policy using `dsconfig`

Creates and configure a new client connection policy using `dsconfig`.

About this task

To create and configure a client connection policy:

Steps

1. Run the `dsconfig` tool.

   Example:

   ```
   $ bin/dsconfig
   ```

2. Enter the connection parameters to the server.

Connection parameters can include:

- Host name
- Connection method
- Port
- bind distinguished name (DN)
- Bind DN password

3. In the PingDirectoryProxy main menu, enter the selection for `Client connection policy configuration`. To create a new client connection policy, enter `2`.

Example:

```
>>>> 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. To create a new client connection policy from scratch, enter `n`.

Example:

```
>>>> 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.

Example:

```
Enter the 'policy-id' for the Client Connection Policy that you
want to create: new_policy
```

6. To set up the policy to be enabled by default or not, select `true` or `false`.

Example:

```
Select a value for the 'enabled' property:

  1) true
  2) false

  ?) help
  c) cancel
  q) quit

Enter choice [c]: 1
```

7. Enter a value for the `evaluation-order-index` property.

> **Note**
>
> Client connection policies with a lower index are evaluated before those with a higher index.

Example:

```
Enter a value for the 'evaluation-order-index' property: 2
```

Result:

The properties of your new client connection policy are displayed.

8. (Optional) To make any further modifications, enter the number corresponding to the property.
9. To finish the creation of the client connection policy, enter `f`.

> **Note**
>
> Any changes that you make to the client connection policy don't apply to existing connections. They only apply to new connections.

Configuring globally unique attributes

PingDirectoryProxy supports a globally unique attributes feature that ensures uniqueness for values defined for a set of attributes within a subtree view.

You can configure the attributes when the server checks for attribute conflicts, either before any `add`, `modify`, or `modifyDN` change request (pre-commit) or after the successful completion of a change request (post-commit).

About the Globally Unique Attribute plugin

The PingDirectoryProxy 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 `modifyDN` 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, such as **`add`**, **`modify`**, or **`modifyDN`**. If a conflict is detected, the change request is rejected. If the `post-commit-validation` property is enabled, the server performs one or more searches after the change has been processed to determine if a conflict was created in multiple servers at the same time. If a conflict is detected in this manner, an administrative alert is generated to notify administrators of the problem so that they can take any manual corrective action.

The Globally Unique Attribute plugin attempts to detect and prevent unique attribute conflicts for changes processed through this PingDirectoryProxy server, but it can't detect conflicts introduced by changes applied by clients communicating directly with backend servers.

You should enable the plugin 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 might not be sufficient for cases in which the content is split across multiple sets of servers, such as in an entry-balanced environment or in proxy configurations with different branches on different servers.

The LDAP SDK uniqueness request control can be used to enforce uniqueness on a per-request basis. Refer to the LDAP SDK documentation and the `com.unboundid.ldap.sdk.unboundidds.controls.UniquenessResponseControl` class for using the control. Also, refer to the ASN.1 specification to implement support for it in other APIs.

Note

Consider 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 increases the processing time for **`add`**, **`modify`**, or **`modifyDN`** operations because the necessary searches to look for conflicts happen before the update request is forwarded to any backend servers.
- Post-commit validation only sends administrative alerts 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 doesn't affect the processing time for the associated write operation.

In most cases, pre-commit validation should be sufficient to prevent conflicts. However, you should periodically run the **`identify-unique-attribute-conflicts`** tool to find any conflicts that might have arisen.

Note

If you want to mitigate any risks caused by 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.

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 might 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, such as `add`, `modify`, or `modifyDN`. If a conflict is detected, the change request is rejected.

> **Note**
>
> You should index all configured attributes for equality in all backend servers.

Steps

- To create the Globally Unique Attribute plugin, run `dsconfig` with the `create-plugin` option.

  Example:

```
$ 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

PingDirectoryProxy supports a global referential integrity plugin mechanism that maintains distinguished name (DN) references from a specified set of attributes to entries that exist in the server, such as between the members values of a static group and the corresponding user entries.

The plugin intercepts `delete` and `modifyDN` operations and updates any references to the target entry. For a `delete` operation, any references to the target entry are removed. For `modifyDN` 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 doesn't have an asynchronous mode. When enabled on the PingDirectoryProxy server, the client response is delayed until the referential integrity processing is complete.

For PingDirectoryProxy server deployments not using entry balancing and using directory server external servers, use the Referential Integrity plugin on the directory server.

> **Note**
>
> 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

- To configure the Global Referential Integrity plugin, run `dsconfig` with the `create-plugin` option.

> **Note**
>
> Any attributes for which referential integrity should be maintained should have values that are DNs and should be indexed for equality in all backend servers.

Example:

In this example, the plugin ensures that the `member`, `uniqueMember`, and `manager` attributes maintain their DN references in the defined subtree views.

```
$ 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

To configure an Active Directory (AD) server backend, run a `dsconfig` script.

The following settings are required for an Active Directory server:

- **`verify-credentials-method:bind-on-existing-connections` and `authorization-method:rebind`**

  > **Note**
  >
  > Active Directory does not support **`proxy-as`**. Existing connections must be reused.

- **`set max-connection-age:5m` and `health-check-pooled-connections:true`**

  > **Note**
  >
  > Active Directory drops idle connections after 15 minutes. The proxy must refresh the connection pool in a shorter interval.

Example

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
```

Deploying a standard PingDirectoryProxy server

You can deploy a PingDirectoryProxy server in a variety of ways depending on the needs of your enterprise.

This section describes and illustrates a standard deployment scenario and includes the following topics:

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-primary 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. 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.

Before 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 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's deployed in an environment. For more information on server profiles, see Server profiles.

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 8.3 supports the use of information in the topology registry to automatically discover the backend PingDirectory server instances to which requests might 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 PingDirectory server instances and only if the following are true:

- The PingDirectoryProxy server must be a member of the same topology as the PingDirectory 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 PingDirectory 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 PingDirectory 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 don't 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 8.0.0.0 supports the addition of PingDirectoryProxy server instances to the same topology as the PingDirectory server instances.

You can do this 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. You can do this 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 Directory 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 walks 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, as shown in the following 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:31:13:18:31:ca
     -
     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:f0:89:a3

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 PingDirectory server topology instead. If you enter `yes`, the process is basically the same as joining an existing directory server topology, and you are prompted for the information needed to connect and authenticate to a PingDirectory server instance in the topology. The primary difference is that you have to define the PingDirectoryProxy server configuration yourself, as shown in the following 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:c0:b8:b7
     -
     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:d7:0d:43:ba

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

About this task

Interactive mode is a convenient method to get the server up and running when you're 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 several useful arguments.

To join a topology with non-interactive setup:

Steps

- Run `setup`:

  Choose from:

  - Use the following arguments to join an existing PingDirectory server topology:

    `--existingDSTopologyHostName {address}`

    The address of a PingDirectory server instance in the topology to be joined.

    `--existingDSTopologyPort {port}`

    The port for communication with the PingDirectory server to retrieve information about the topology.

    `--existingDSTopologyUseSSL`

    Indicates that the communication with the PingDirectory server to retrieve information about the topology should be encrypted with SSL.

    `--existingDSTopologyUseStartTLS`

    Indicates that the communication with the PingDirectory server to retrieve information about the topology should be encrypted with the StartTLS extended operation.

    `--existingDSTopologyUseNoSecurity`

    Indicates that the communication with the PingDirectory server to retrieve information about the topology shouldn't be encrypted.

    `--existingDSTopologyUseJavaTruststore{path}`

    The path to a JKS trust store that has the information needed to trust the certificate presented by the PingDirectory 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 PingDirectory 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 PingDirectory 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 PingDirectory server.

--existingDSTopologyBindPasswordFile{path}

The path to a file containing the password to use to authenticate to the PingDirectory 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 PingDirectory 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
```

- Use the following arguments to clone the configuration of an existing PingDirectoryProxy server instance, including joining the same topology as the existing instance:

`--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 shouldn't 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

About this task

You can use the `manage-profile` tool to set up an instance of the server from information contained in a server profile. This tool invokes `setup` and performs other tasks, such as applying configuration changes, installing schema and extensions, and adding files to the server root.

Steps

1. Place an appropriate set of arguments 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`.

   Because `manage-profile setup` uses the setup tool in non-interactive mode, you should use the arguments listed in the previous section, including:

   - `--existingDSTopologyHostName {address}`
   - `--existingDSTopologyPort {port}`
   - `--existingDSTopologyUseSSL`
   - `--existingDSTopologyUseStartTLS`
   - `--existingDSTopologyUseNoSecurity`
   - `--existingDSTopologyUseJavaTruststore {path}`
   - `--existingDSTopologyUsePkcs12Truststore {path}`
   - `--existingDSTopologyTrustStorePassword {password}`

- `--existingDSTopologyTrustStorePasswordFile {path}`
- `--existingDSTopologyBindDN {path}`
- `--existingDSTopologyBindPassword {password}`
- `--existingDSTopologyBindPasswordFile {path}`

> **Note**
>
> Unlike the **`setup`** utility, **`manage-profile setup`** doesn't support cloning an existing PingDirectoryProxy server instance, so the `--peerHostName`, `--peerPort`, and other related arguments can't be included in the `setup-arguments.txt` file.

2. Run **`manage-profile setup`**.
3. If you have already set up an instance of the server, 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 includes an appropriate set of arguments for joining the same topology.

Joining a topology with manage-topology add-server

Steps

- 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 isn't already part of any other topology, since it's 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 the following:

- `useSSL` indicates that the communication should be encrypted with SSL.
- `useStartTLS` indicates that the communication should be encrypted with the StartTLS extended operation.
- `noSecurity` indicates that the communication shouldn't 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.

Example:

Use a command similar to the following to add a PingDirectoryProxy server to an existing PingDirectory 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`

Additionally, the `health-check-state` property is not available for LDAP external server templates because 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 PingDirectory servers, most of the remaining properties in LDAP external server templates have the same default values as the corresponding properties in the PingDirectory server external server type. However, there are 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 PingDirectory server external servers. The `inter-server` authentication type indicates that the PingDirectoryProxy server should authenticate to the PingDirectory 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 PingDirectory server instances are 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 PingDirectory 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 PingDirectory 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 might want to add custom health checks that are invoked against servers created from the template. The PingDirectoryProxy server automatically checks to see whether the server reports any degraded or unavailable alert types, and also verifies that the backend server's root DSA-specific entry (DSE) is accessible in a timely manner, but you might want 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 earlier than 8.0.0.0, each load-balancing algorithm had to be explicitly configured with a set of backend servers. This is still supported and might be the preferred configuration in some cases.

For example, explicitly configured algorithms are required if any backend server is not a PingDirectory server or if it is a PingDirectory server version earlier than 8.0.0.0. It might also be preferable if you want to have more direct control over the configuration that the PingDirectoryProxy server uses for each backend server.

Beginning with 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 automatically discovers its backend servers, as in the following example.

```
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

With the failover load-balancing algorithm, the backend servers are ordered lexicographically by instance name. This ensures that all PingDirectoryProxy server instances 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 because 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 want to direct all traffic to a specific instance as long as it is available, then it might 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, you should use both of these together with failover load-balancing for writes and fewest operations for reads. When combined with the PingDirectory 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 the PingDirectory server is configured with an LDAP external server template, either the fewest operations algorithm or the failover load-balancing algorithm uses the topology registry to determine which PingDirectory 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 PingDirectory 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:fewest-operations-load-balancer \
     --add load-balancing-algorithm-name:failover-load-balancer
```

The PingDirectoryProxy Server automatically detects and reacts to applicable changes in the topology registry. This includes the following:

Adding a new PingDirectory 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 PingDirectory server instance

If this instance was associated with any load-balancing algorithms, then it is no longer eligible to receive requests through these load-balancing algorithms.

Modifying an existing PingDirectory 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 PingDirectory 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 might optionally include criteria-based load balancing. Each of these load-balancing algorithms can 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 can be automatically determined from the information in the topology registry.

Creating a standard multi-location deployment

In this example deployment, the 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 PingDirectoryProxy 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 distinguished name (DN) enabled:

- `ds-east-01.example.com`
- `ds-east-02.example.com`
- `ds-west-01.example.com`
- `ds-west-01.example.com`

You will configure four PingDirectoryProxy 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

This deployment scenario involves the following procedure.

- Install the first PingDirectoryProxy server in the 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 that 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 PingDirectoryProxy server using the configuration cloning feature.
- Install two PingDirectoryProxy server instances in the west location, which includes using the `setup` file and manually setting the location to west using the `dsconfig` command and then copying the configuration of the PingDirectoryProxy server using the configuration cloning feature.

After you have configured and tested the proxy server, tour the configuration of each of the proxy server components. You can modify these properties later as needed using the `dsconfig` tool.

Installing the first PingDirectoryProxy server

Install the first PingDirectoryProxy server from the `.zip` installation file.

About this task

To begin with, we have the PingDirectoryProxy installation `.zip` file. In this example, we plan to use SSL security, so we also have a key store certificate database and a `.pin` file that contains the private key password for the key store. The key store files are only necessary when using SSL or StartTLS.

In this deployment scenario, the key store database is assumed to be a Java KeyStore (JKS), which can be created by the keytool program.

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` key store file contains the private key entry for the `proxy-east-01.example.com` server certificate with the alias `server-cert`. The server certificate, certificate authority (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 this example. The private key password in this example is the same as the password defined for the `ExampleKeystore.jks` key store.

Steps

1. Extract the compressed archive file into the PingDirectoryProxy directory and change to this directory.

   Example:

   ```
   root@proxy-east-01: unzip -q PingDirectoryProxy-<version>-with-je.zip
   root@proxy-east-01: cd PingDirectoryProxy
   ```

2. Copy the key store and `.pin` files into the `config` directory.

   Example:

   ```
   root@proxy-east01: cp ../Keystore config/
   root@proxy-east01: cp ../Truststore config/
   ```

3. Install the first proxy server by running the `setup` tool on `proxy-east-01.example.com`.

   Example:

   ```
   root@proxy-east01: ./setup --no-prompt --acceptLicense \
   --ldapPort 389 --rootUserPassword pass \
   --maxHeapSize 1g --enableStartTLS --ldapsPort 636 \
   --useJavaKeystore config/ExampleKeystore.jks \
   --keyStorePasswordFile config/ExampleKeystore.pin \
   --certNickname server-cert \
   --useJavaTrustStore config/ExampleTruststore.jks
   ```

   Result:

   New key store password files are created in `config/keystore.pin`. The original file, `config/ExampleKeystore.pin`, is no longer needed. If you are not using SSL or StartTLS, then the SSL arguments are not necessary.

   ```
   root@proxy-east01: ./setup --no-prompt --acceptLicense \
   --ldapPort 389 --rootUserPassword pass --maxHeapSize 1g
   ```

Next steps

After installing the PingDirectoryProxy server, you can configure it using the `create-initial-proxy-config` tool as presented in Configuring the first PingDirectory server.

Configuring the first PingDirectoryProxy server

After the PingDirectoryProxy server has been installed, it can be automatically configured using the `create-initial-proxy-config` tool.

About this task

The `create-initial-proxy-config` tool can only be used once for this initial configuration, after which you must use `dsconfig` to make any changes to your proxy server configuration.

Configuring the PingDirectoryProxy server with the `create-initial-proxy-config` tool involves the following steps:

- Providing PingDirectoryProxy server base distinguished name (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 PingDirectoryProxy server.

Steps

1. After completing setup, run the `create-initial-proxy-config` tool.

   Example:

   ```
   root@proxy-east01: bin/create-initial-proxy-config
   ```

2. Provide the bind DN and password that the PingDirectoryProxy 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 PingDirectoryProxy 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 PingDirectoryProxy 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 you are only using one proxy base DN, press Enter to finish.

Defining locations

Define the first location, east, to accommodate the servers in the deployment located on the East Coast of the United States.

About this task

Continuing from the same `create-initial-proxy-config` session:

Steps

1. Enter a location name for the PingDirectoryProxy server. Press Enter.

   In this example, enter `east`.

2. Define a location named `west` for the servers in our deployment located on the West Coast. Press Enter.

3. Select the location that contains the PingDirectoryProxy server itself.

   The PingDirectoryProxy server is located in the east.

Configuring the external servers in the east and west locations

After the locations have been defined, identify the directory servers.

Start by defining 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.

   Example:

   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 setting up the `cn=Proxy User` account on the PingDirectoryProxy 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.

Apply the configuration to the PingDirectoryProxy server

Review the configuration summary.

About this task

After confirming that the changes are correct, press Enter to write the configuration.

Steps

1. Review the configuration and then apply the changes to the PingDirectoryProxy server. Press Enter to write the configuration to the server.

   During the configuration process, the **`create-initial-proxy-config`** tool writes the configuration settings to a **`dsconfig`** batch file, which are applied to the PingDirectoryProxy 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.

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.

   After the changes have been applied, the **`create-initial-proxy-config`** tool cannot be used to configure this proxy server again.

Configuring additional PingDirectoryProxy server instances

Install and configure additional PingDirectoryProxy server using the **`setup`** tool.

About this task

Install and configure the second PingDirectoryProxy server by running the **`setup`** tool on `proxy-east-02.example.com`.

Steps

1. Copy the key store and `.pin` files into the `config` directory for the `proxy-east-02.example.com` server.

   Example:

   ```
   root@proxy-east-02: cp ../Keystore config/
   root@proxy-east-02: cp ../Truststore config/
   ```

2. Install the second PingDirectoryProxy server by running the `setup` tool on `proxy-east-02.example.com`.

   Example:

   ```
   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 \
   --maxHeapSize 1g --localHostName proxy-east-02.example.com \
   --peerHostName proxy-east-01.example.com \
   --peerPort 389 --location east
   ```

3. Configure the third PingDirectoryProxy server, `proxy-west-01.example.com` in the same way as shown in steps 1 - 2.

   Example:

   ```
   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`.

   Example:

   ```
   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 \
   --maxHeapSize 1g --localHostName proxy-west-01.example.com \
   --peerHostName proxy-east-01.example.com \
   --peerPort 389 --location west
   ```

5. Repeat steps 1 - 4 to install the last PingDirectoryProxy server.

Result

All proxies have the same Admin Data backend and have the `all-servers` group defined as their configuration-server-group in the PingDirectoryProxy Server Global Configuration object. When making a change to a PingDirectoryProxy server using the `dsconfig` command-line tool or the admin console, you have the choice to apply the changes locally only or to all proxies in the `all-servers` group.

Testing external server communications after initial setup

After setting up the basic deployment scenario, the communication between the proxies and the LDAP external servers can be tested using a feature in the proxy server in combination with an LDAP search.

About this task

After initial setup, the PingDirectoryProxy server exposes a special search base distinguished name (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`.

Steps

1. Run `dsconfig` to set the `include-backend-server-passthrough-subtree-views` property to `TRUE`.

   Example:

   ```
   root@proxy-east-01: dsconfig set-client-connection-policy-prop \
   --policy-name default \
   --set include-backend-server-passthrough-subtree-views:true
   ```

   Result:

   When set to `TRUE`, an LDAP search against the PingDirectoryProxy server with the base DN `dc=example,dc=com,ds-backend-server=ds-east-02.example.com:389` instructs the PingDirectoryProxy 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 might 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.

   Example:

   ```
   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. 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.

   Example:

   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

After you have tested connectivity, run a simulated failure of a load-balanced external server to verify that the PingDirectoryProxy server redirects LDAP requests appropriately.

About this task

In this procedure, stop the `ds-east-01.example.com:389` server instance and test searches through `proxy-east-01.example.com`.

Steps

1. Perform several searches against the PingDirectoryProxy 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 you used the default load balancing algorithm of fewest operations, it's likely that all of the searches go to only one of the proxies.

   Example:

   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. Stop the directory server instance on `ds-east-01.example.com` using the `stop-server` command and immediately retry the searches in step 1.

   > **Note**
   >
   > There should be no errors or noticeable delay in processing the search.

   Example:

   ```
   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 PingDirectoryProxy server instance on `ds-east-01.example.com`.

4. Check the access log to confirm that the PingDirectoryProxy server started to include the `ds-east-01` server in load-balancing within 30 seconds.

   The default time is 30 seconds, but you can change this default.

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 use StartTLS to communicate with the directory servers in the central region.

> **Note**
>
> Other than the ability to add to the PingDirectoryProxy server's trust store, the **`prepare-external-server`** tool does not alter the PingDirectoryProxy server configuration in any way.

The PingDirectoryProxy server itself, installed on `proxy-east-01.example.com`, remains in the east location. This example reconfigures load balancing between the six directory servers in these 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

This deployment scenario takes the following steps:

1. **Prepare the new external servers using the `prepare-external-server` tool.**
2. **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.**
3. **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

Prepare the external directory servers, `ds-central-01` and `ds-central-02`, by creating the proxy user account and the supporting access rules.

About this task

Connect to the `ds-central-01` PingDirectory server using StartTLS. Because you are using StartTLS, you must capture the `ds-central-01` server's certificate and put it in the trust store on your PingDirectoryProxy server instance.

The `prepare-external-server` tool is located in the `bin` or `bat` directory of the server root directory, PingDirectoryProxy. In this example, run the tool on the `ds-east-01` instance of the PingDirectoryProxy server.

Steps

1. **Run the `prepare-external-server` tool to prepare the two new servers.**

   Example:

On the first attempted bind to the server, the tool reports a `failed to bind` message because it can't bind to the `cn=Proxy User` entry because it hasn't been created yet. The tool sets up the `cn=Proxy User` entry so that the PingDirectoryProxy 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 distinguished name (DN) is required when using **`prepare-external-server`**.

Adding the new PingDirectory servers to the PingDirectoryProxy server

After preparing the external PingDirectory servers to communicate with the PingDirectoryProxy server, add the two servers in the central location to the proxy server instance.

About this task

Because you ran the `prepare-external-server` tool, the two servers have the `cn=Proxy User` entry configured.

Steps

- Run the `dsconfig` tool.

  Example:

```
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

Add a new central location to which the new PingDirectory servers will be added.

About this task

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.

   Example:

   ```
   $ bin/dsconfig
   ```

2. In the main menu, enter the number corresponding to Location.
3. In 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 if 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

Edit the existing east and west locations to include the new central location in their failover logic.

About this task

The new failover logic is be based on geographic distance so that the east location will first fail over to the 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. In the PingDirectoryProxy server console configuration menu, enter the number corresponding to Location.
3. In 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.

   You will add the west location 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

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.

   Example:

   As this health check is for the `ds-central-01` server, the name takes the following format.

```
>>>> Enter a name for the Search LDAP Health Check that you want to create:
ds-central-01.example.com:389_dc_example_dc_com-search-health-check
```

6. Review the configuration properties and then enter `f` to finish configuring the new health check and save changes.
7. Repeat steps 2-6 to create another new health check for the `ds-central-02` server.

Adding new external servers

Add new external servers using `dsconfig`.

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. In 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.

   Example:

   ```
   >>>> 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. In the `External Server menu`, change the `server-host-name` property to reflect the name of the `ds-central-01` server.
9. In 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 steps 1-15 to add the new `ds-central-02` external server.

Modifying the load-balancing algorithm

Modify the existing load-balancing algorithm to include the newly created servers.

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. Select 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.

   For 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.

   Result:

   The change is saved and applied to the PingDirectoryProxy server. The load-balancing algorithm is referenced in the `load-balancing-algorithm` property of the request processor used by this PingDirectoryProxy server.
8. To view this property, go to the main menu and select the `Request Processor` option.
9. In the `Request Processor menu`, enter the number corresponding to view and edit an existing request processor.
10. Select the request process used by the PingDirectoryProxy server, and review the configuration properties.

    Result:

    This request processor is used by the subtree view serviced by the PingDirectoryProxy server, which is in turn referenced by the client connection policy.

> **Note**
>
> The changes made in this procedure are already in effect. The PingDirectoryProxy server does not have to be restarted.

Testing external server communications after initial setup

After setting up the basic deployment scenario, the communication between the proxies and the LDAP external servers can be tested using a feature in the proxy server in combination with an LDAP search.

About this task

After initial setup, the PingDirectoryProxy server exposes a special search base distinguished name (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`.

Steps

1. Run `dsconfig` to set the `include-backend-server-passthrough-subtree-views` property to `TRUE`.

   Example:

   ```
   root@proxy-east-01: dsconfig set-client-connection-policy-prop \
   --policy-name default \
   --set include-backend-server-passthrough-subtree-views:true
   ```

   Result:

   When set to `TRUE`, an LDAP search against the PingDirectoryProxy server with the base DN `dc=example,dc=com,ds-backend-server=ds-east-02.example.com:389` instructs the PingDirectoryProxy 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 might 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.

   Example:

   ```
   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. 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.

   Example:

   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

After you have tested connectivity, run a simulated failure of a load-balanced external server to verify that the PingDirectoryProxy server redirects LDAP requests appropriately.

About this task

To run a simulated external server failure:

Steps

1. 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 PingDirectoryProxy server and verify activity in each of the servers in the east location, `ds-east-01` and `ds-east-02`, by looking at the access logs.

   Example:

   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. 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 searches in step 2.

   There should be no errors or noticeable delay in processing the search.

   Example:

   ```
   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 because 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`.
6. 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 Corporation's `o=sample` rooted directory, converting Sample Corporation'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 Corporation's directory clients to become aware of the new directory information tree (DIT) and schema, proxy data transformations are created to give the Sample Corporation's 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 Corporation schema.

Overview of the attribute and DN mapping

To achieve the merger of the two data sets, create proxy transformations that map the Sample Corporation's source attributes to Example.com target attributes.

The Example.com schema already defines an attribute to contain the relative distinguished name (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 Corporation schema for representing the region and the distinguished name (DN) of linked accounts.

During the merger, Example.com decides to re-parent the Sample Corporation'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 the DN Mappings table below. In this process, Example.com collapses the Sample Corporation tree, moving entries from the east and west region under a single DN, `dc=example,dc=com`. The DN proxy transformations assume that all the Sample Corporation users have been co-located under this single Example.com subtree.

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 Corporation 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 Corporation legacy applications.

Because the Example.com directory server has no knowledge of the Sample Corporation user's former `ou=east` or `ou=west` association, search results for client searching under `o=sample` will return a DN that might differ from the original search base. For instance, a search for `sampleID=abc123` under `ou=west,o=sample` might return the user entry for abc123 with the DN of `sampleID=abc123,ou=east,o=sample`.

The following table illustrates the mapping DNs.

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 distinguished names (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 might 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 you can create separate subtree views in the same client connection policy. For example, you would create one subtree view 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 is an example of a Sample Corporation 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.

> **Note**
>
> This view is how the entry appears to search requests under the `dc=example,dc=com` base distinguished name (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 example shows what the PingDirectoryProxy server returns to LDAP clients who have requested the entry when searching under the `o=sample` base DN.

> **Note**
>
> The DN returned includes `ou=east` even though this branch doesn't exist in the Example.com directory information tree (DIT). It also returns the attribute names as they are defined in the Sample Corporation 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

This deployment scenario takes the following steps:

- Install any necessary schema on the PingDirectoryProxy server.
- Create three attribute mapping proxy transformations and three distinguished name (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 the configuration by performing some searches on the Sample Corporation directory information tree (DIT).

About the schema

The PingDirectoryProxy server inherits user-defined schema from all external servers by comparing `cn=schema` on these servers at PingDirectoryProxy server startup and at five minute intervals.

In the following example, the Example.com company acquires Sample Corporation. The Example.com schema does not need to be added manually to the PingDirectoryProxy server's `config/schema` directory. We assume that the schema for Sample Corporation's entries has been defined on the external servers with the example.com directory information tree (DIT), requiring no direct schema management on the PingDirectoryProxy 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 Corporation schema, such as `sampleRegion` and `sampleID`.

Creating proxy transformations

In this deployment scenario, you create three attribute mapping proxy transformations and three distinguished name (DN) mapping proxy transformations.

About this task

To create attribute and distinguished name (DN) mapping proxy transformations, you can use 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.
2. Run `dsconfig` in interactive mode and enter the LDAP connection parameters.
3. Inn the `Configuration main menu`, enter the number corresponding to your desired `Proxy Transformation`.

Creating the Attribute Mapping Proxy Transformations

Create the attribute mapping proxy transformations using `dsconfig` interactive.

For this example, assume you are continuing from the previous `dsconfig` session in Creating proxy transformations. In Creating the DN mapping proxy transformations, this transformation maps `ou=east,o=sample` in the `Sample Corporation` schema `dc=example,dc=com` in the `Example.com` schema.

Creating the DN mapping proxy transformations

Create the distinguished name (DN) mapping proxy transformations.

Steps

1. In the `Proxy Transformation` menu, enter the number corresponding to `Create a new Proxy Transformation`.
2. Enter the option to create a new `Proxy Transformation`.
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 Corporation` schema `dc=example,dc=com` in the `Example.com` schema.
5. To enable the transformation by default, select `True`.
6. Specify the source DN as it appears in client requests.

   Example:

```
>>>> 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.

   Example:

```
>>>> 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. To create the new DN mapping proxy transformation, enter `f`.
9. Repeat the previous steps to create a new DN mapping proxy transformation that maps `ou=west,o=sample` in the `Sample Corporation` schema to `dc=example,dc=com` in the `Example.com` schema, and name it `sample_west-to-example`.
10. Create a DN mapping proxy transformation for the base DN of the `Sample Corporation` database.

Creating a request processor to manage the proxy transformations

Create a new proxying request processor that includes your new attribute and distinguished name (DN) mapping proxy transformations.

About this task

Use the existing `dc_example_dc_com` request processor as a template.

Steps

1. In 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. Select the option to use the current request processor as a template.
4. Enter a name for the new proxying request processor, such as `o_sample-req-processor`.
5. Review the properties and enter the number corresponding to the `Transformation` property.

> **Note**
>
> The load-balancing algorithm is the same as for the previous request processor, but you must change the transformation property.

6. Enter the number corresponding to the proxy transformations that were previously created.
7. Select the attribute mapping proxy transformations, then select the DN mapping proxy transformations.

> **Note**
>
> The order of the selection is important because you have related DNs. Begin with the DNs that are lower in the tree, and finish with the base DN transformation.

Example:

```
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. To save your changes, enter `f`.

Creating subtree views

About this task

To configure subtree views for the PingDirectoryProxy server:

Steps

1. In the `Configuration` main menu, enter the number corresponding to `Subtree View`.
2. In 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 subtree.
4. Select the `dc_example_dc_com-view` subtree view.
5. Enter a descriptive name for the subtree view configuration.
6. Configure the base distinguished name (DN) property of the `Sample Corporation` data set.
7. Enter the request processor created in Creating a request processor to manage the proxy transformations.
8. To save the changes, press `f`.

   Example:

```
>>>> 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

Edit the client connection policy.

About this task

To add the new `o=sample` subtree view:

Steps

1. In the `Configuration` main menu, enter the number corresponding to `Client Connection Policy`.
2. In the `Client Connection` menu, enter the number corresponding to `Create a new Client Connection`.
3. In the configuration properties, select the `subtree-view` property and 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 you created in Creating subtree views.

   Example:

   ```
   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

You can test the proxy transformation service by imitating example client requests to search and modify users.

About this task

After setting up the deployment scenario, the PingDirectoryProxy server will respond to requests to the `dc=example,dc=com` and `o=sample` base distinguished names (DNs).

Steps

1. To view a Sample Corporation entry, run `ldapsearch`.

   Example:

   The following example fetches the user with `sampleID=scase` under the `ou=east,o=sample` base DN.

   ```
   root@proxy-east-01: bin/ldapsearch --bindDN "cn=directory manager" \
   --bindPassword password --baseDN "ou=east,o=sample" "(sampleID=scase)"
   ```

   Result:

   ```
   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 by changing it to `west`.

   Example:

   To modify the value, create an `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.

   Example:

```
root@proxy-east-01: bin/ldapmodify --bindDN "cn=Directory Manager" \
--bindPassword password --filename scase-mod.ldif
```

Result:

```
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 the `scase` `sampleRegion` value under `o=sample`

   Example:

   ```
   root@proxy-east-01: bin/ldapsearch --bindDN "cn=directory manager" \
   --bindPassword password --baseDN "o=sample"  "(sampleID=scase)" \
   sampleRegion
   ```

   Result:

   ```
   dn: sampleID=scase,ou=People,ou=east,o=sample
   sampleRegion: west
   ```

5. To search for `scase`, use `uid` instead of `sampleID`, under the `dc=example,dc=com` base DN.

   Example:

   You can see the Example.com schema version of the entry in the following sample.

   ```
   root@proxy-east-01: bin/ldapsearch --bindDN "cn=directory manager" \
   --bindPassword password --baseDN "dc=example,dc=com"  "(uid=scase)"
   ```

   Result:

   ```
   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 PingDirectoryProxy server

Depending upon the needs of your enterprise, you can deploy a PingDirectoryProxy server in a variety of ways.

This section describes and illustrates an entry-balancing deployment scenario that involves:

- Deploying an entry-balancing proxy configuration
- Rebalancing your entries
- Managing the global indexes in entry-balancing configurations
- Working with alternate authorization identities

Deploying an entry-balancing proxy configuration

Entry-balancing is a PingDirectoryProxy server configuration that allows the entries within a portion of the directory information tree (DIT) to reside on multiple external servers.

This configuration is 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 distinguished name (DN) to be divided among any number of separate directory servers, making the PingDirectoryProxy 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 PingDirectoryProxy server. You must properly configure replication on the external directory servers 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 PingDirectoryProxy server communicates with four external directory servers. The PingDirectoryProxy 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 external servers describes the replication configuration that exists before creating the PingDirectoryProxy 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 PingDirectory server instance can hold all of your data, then you should store your data on a single server and replicate for high availability to simplify your deployment.

If a single server can't hold all of your data, then you can spread it across multiple servers in these 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 PingDirectory server instance can hold all of the information within one or more branches. Configure the PingDirectoryProxy server with multiple base distinguished names (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 might be spread across multiple servers. Any entries that are further subordinates must be placed in the same directory server instance as their parent.
- If one or more branches are so large that any single PingDirectory server instance can't hold all of the data, you need to use entry balancing within that branch to divide the entries among two or more sets of PingDirectory servers. You might also need to change the way that the data is arranged in the server so that it uses as flat a directory information tree (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 PingDirectoryProxy 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`.

> **Note**
>
> 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 PingDirectory server data.

When defining access controls in an entry-balancing deployment, you must ensure that the data used by the access control rule is available for evaluation on all data sets.

If you use groups for access control and a group contains users from different data sets, then that group must exist on each data set. For a single ACI to apply to entries in all data sets, 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 data sets.
- 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 PingDirectoryProxy 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 is kept in sync if changes are made through the PingDirectoryProxy server.

Overview of deployment steps

This deployment scenario takes the following steps.

- Install the PingDirectoryProxy 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 PingDirectoryProxy server

About this task

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.

  Example:

```
root@austin-proxy1: ./setup --acceptLicense \
--listenAddress austin-proxy1.example.com \
--ldapPort 389 --rootUserDN "cn=Directory Manager" \
--rootUserPassword pass --entryBalancing \
--maxHeapSize 2g --no-prompt
```

Configuring the entry-balancing PingDirectoryProxy server

After the PingDirectoryProxy server has been installed, it can be automatically configured using the `create-initial-proxy-config` tool.

About this task

This tool can only be used once for this initial configuration after which you will have to use `dsconfig` to make any changes to the PingDirectoryProxy server configuration.

Steps

1. Run the `create-initial-proxy-config` tool.

   Example:

```
root@austin-proxy1: ./bin/create-initial-proxy-config
```

2. If the topology meets the requirements, press Enter to continue.

   Example:

```
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 the proxies have identical proxy user accounts and passwords.

   Example:

```
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 PingDirectoryProxy server will use to communicate with PingDirectory servers.
5. Enter a base distinguished name (DN) of the PingDirectory server instances that will be accessed by the PingDirectoryProxy server.
6. Define the balancing point as a separate base DN, which is entry balanced.

   Example:

```
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. Because 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.

   Example:

```
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. Because the balancing point is the same as the base DN, `ou=people,dc=example,dc=com`, use it as the entry balancing base.

   Example:

```
>>>> 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.

    Example:

    ```
    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 PingDirectoryProxy 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 relative distinguished name (RDN) index priming.

    Example:

    ```
    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.

    Example:

```
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. Define austin as the location of this PingDirectoryProxy server instance.

    Example:

```
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.

    Example:

```
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 PingDirectoryProxy server instance.

    Example:

```
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.

    Example:

```
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.

    Example:

```
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.

    Example:

```
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.

Example:

```
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 PingDirectoryProxy server, specifying the directory manager password for the initial creation of the proxy user.

    Example:

```
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. Because the replication set name has already been configured, you do not need to use the name created automatically by the PingDirectoryProxy server.

    Example:

```
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 in steps 17-24.

    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. To apply the configuration changes to the PingDirectoryProxy server, enter `yes`.

    Example:

```
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

Configure the placement algorithm using a batch file.

About this task

You want to place new entries added through the proxy using LDAP `ADD` operations into the least used dataset. Do this using an entry-count placement algorithm. To change the placement algorithm from round-robin to entry-count, first create and enable an entry-count placement algorithm configuration object, and then disable the existing round-robin placement algorithm. The 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**
>
> 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.

  Example:

  ```
  root@austin-proxy1: bin/dsconfig --no-prompt \
  --bindDN "cn=directory manager" --bindPassword password \
  --port 389  --batch-file ../dsconfig.post-setup
  ```

  Result:

  After 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.

```
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 third party algorithm, you might need to redistribute your entries.

If 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. If the backends all reach their maximum capacity and you decide to add a new backend to the deployment, you must 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, such as by 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.

This section describes each of these methods of entry rebalancing in more detail.

About dynamic rebalancing

During dynamic rebalancing, entries get moved as they are modified. You configure dynamic rebalancing in 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. 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 might 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`

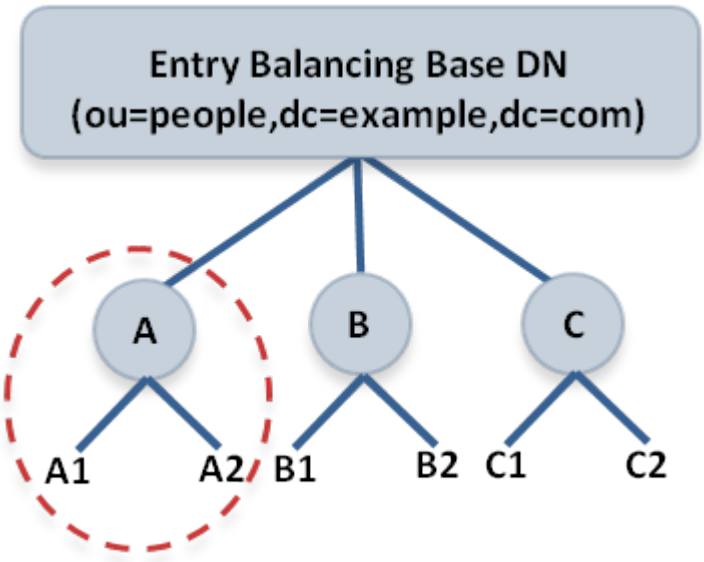
Specifies the maximum size of a subtree that can be rebalanced. `poll-interval`

Specifies how long to wait between polling the size of the backends to determine how to rebalance and 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`
- `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`.



If you are writing your own third-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

Configure dynamic rebalancing on an existing entry balancing configuration.

Steps

1. If the current placement algorithm does not support rebalancing, create an entry counter placement algorithm.

   Example:

   You can either do this using `dsconfig` in interactive mode or using the `dsconfig` command line as in this example.

   ```
   $ 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. Throttle how many entries are being moved by the proxy so that the backend servers do not have too heavy a load by setting the `rebalancing-queue-maximum-size` property of the request processor created in the previous step.

   Example:

   By default, the value is 1000. If the load is too high, reduce this value

```
$ 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. 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.

   Example:

   This example configures the access log to display entry rebalancing processing information.

```
$ 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 distinguished names (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 might 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. When 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 PingDirectory 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 can't 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 PingDirectoryProxy server maintains the default relative distinguished name (RDN) index as well as one or more in-memory global attribute indexes.

The global indexes allow the PingDirectoryProxy server to select the correct backend server set for incoming operations, which avoids broadcasting operations to all backend sets.

The indexes can 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 PingDirectoryProxy server. For example, when a new entry is added, the distinguished name (DN) of the new entry is added to the DN index of the PingDirectoryProxy server performing the operation. The indexes are also fault-tolerant and can adapt to changes made in the backend servers without going through the PingDirectoryProxy server. For example, operations are 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 the global index growth
- How to prime the global index from a peer at startup

Creating a global attribute index

Create a global attribute index for PingDirectoryProxy server to use as a optional index.

About this task

The relative distinguished name (RDN) index is referenced whenever a `modify`, `delete`, or base search is requested. The RDN index is needed when the LDAP request contains the complete distinguished name (DN) of the targeted entry. If the entry-balancing request processor is not configured to prime the RDZ 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 PingDirectoryProxy server is handling a search request with an equality filter involving an attribute, such as the `telephoneNumber` attribute with the `telephoneNumber=+11234567890` filter.

Because the PingDirectoryProxy server doesn't know what the data within the subtree view looks like or how it is searched, it can't 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 processed by the PingDirectoryProxy server. The PingDirectory server must also have an equality or ordering index type for the associated attribute Local database (DB) Index.

The common candidates for global attribute indexing are the uniquely-valued equality-indexed attributes on the external servers, such as `uid`, `mail` and `telephoneNumber`. These attributes don't require unique values for use as global attribute indexes by the entry-balancing request processor.

Steps

- Create a global attribute index using the `dsconfig` tool with the `create-global-attribute-index` option.

  Example:

  In this example, a PingDirectoryProxy server deployment expects to process frequent searches of the form `(&(mail=user@example.com)(objectclass=person))`. The `mail` attribute is indexed for equality searches on each of the external servers behind the PingDirectoryProxy server. Because the filter is constructed with an equality match and `&` clause, you can use a global attribute index on the `mail` attribute to avoid forwarding the search request to each entry balanced dataset.

  ```
  $ bin/dsconfig create-global-attribute-index \
    --processor-name ou_people_dc_example_dc_com-eb-req-processor \
    --index-name mail --set prime-index:true \
  ```

Result

The index begins populating as search requests are made to the PingDirectoryProxy server that involve the mail attribute.

> **Tip**
>
> You can use the `reload-index` tool to fully populate the index for optimal performance as described in Reloading the global indexes.

Reloading the global indexes

The `reload-index` tool allows you to manually reload the PingDirectoryProxy server global indexes.

Use the `reload-index` tool to reload all configured indexes in the global index, including the relative distinguished name (RDN) index and all attribute indexes or to reload only the indexes you specify.

You might need to reload the index when:

- The PingDirectoryProxy 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.

The `reload-index` tool schedules an operation to run within the PingDirectoryProxy server's process. You must supply the LDAP connection information so that the tool can communicate with the server through its task interface. You can schedule tasks to run immediately or at a later time. After the tasks are scheduled, manage them using the `manage-tasks` tool.

Reloading all of the indexes

About this task

Reload all the indexes within the scope of the base distinguished name (DN) `dc=example,dc=com`. The task is performed as `cn=Directory Manager` on port 389 of the localhost server.

> **Note**
>
> The existing index contents are erased before reloading.

Steps

- To reload all of the indexes within the scope of the `dc=example,dc=com` base DN, run the **`reload-index`** tool.

  Example:

  ```
  $ bin/reload-index --task --bindPassword password --baseDN "dc=example,dc=com"
  ```

Reloading the RDN and UID index

About this task

To reload the RDN and unique identifier (UID) index in the background so that the existing contents of these indexes can continue to be used:

Steps

- Run the **`reload-index`** tool with the `--index rdn` and `--index uid` options.

  Example:

  ```
  $ 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 PingDirectoryProxy server to prime from the backend directory server using the `--fromDS` option, which overrides the configuration of the `prime-index-source` property.

> **Tip**
>
> You can do this on a one-off basis if the global index appears to be growing too large.

To prime the backend server:

Steps

- Run the **`reload-index`** tool with the `--fromDS` option.

Example:

```
$ bin/reload-index --bindPassword password --baseDN "dc=example,dc=com" --fromDS
```

Monitoring the size of the global indexes

Monitor the size of the global indexes over LDAP and reload the indexes to prevent a build up of stale entries in the global indexes.

About this task

The proxies don't communicate changes to the indexes with one another, which can result in stale entries building up over time in the global indexes. In this situation, the PingDirectoryProxy server continues to operate normally because the global indexes are used only as a hint for where to find entries.

The rate of this growth is typically slow because in most environments the key attributes change infrequently. In addition, the global indexes are compact. However, if the global indexes start to fill up the allocated memory, you might need to flush and reload them.

If the global indexes become full, the PingDirectoryProxy server continues to operate normally, but it needs to start evicting entries from the indexes. Evicting the entries leads to more broadcast searches and reduces the overall performance of the PingDirectoryProxy server.

Steps

- To monitor the size of the global indexes over LDAP, use the `ldapsearch` with the `global-index-current-memory-percent` option.

  Example:

  ```
  $ 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
  ```

- To reload the indexes so they no longer hold stale information, run the `reload-index` command with the `--fromDS` option.

  This option ensures that data is loaded from backend directory servers.

  > **Tip**
  >
  > You should reload the indexes during off-peak hours because it might impact the performance while the reload is in progress.

Sizing the global indexes

Use the PingDirectoryProxy server `global-index-size` tool to help you estimate the memory size of your global indexes.

About this task

You can estimate the size of more than one index in a single invocation by providing multiple sets of options.

The `global-index-size` tool creates its estimate using the following information:

Number of keys in the index

For example, for the built-in relative distinguished name (RDN) index, the number of keys is the total number of entries in the PingDirectory server that are immediately below the balancing point. Entries more than one level below the balancing point and entries that are not subordinate to the balancing point aren't contained in the RDN index. For attribute indexes, the number of keys is 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 the attribute value. For the built-in RDN index, the key is the RDN directly below the balancing base distinguished name (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 PingDirectory 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

- To estimate the size of two separate indexes, run the `global-index-size` tool.

  Example:

  In this example, the two indexes both have 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 PingDirectoryProxy server can prime the global indexes at 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 occurs from these sources. 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 is attempted from these sources. Priming is most efficient if the source server is on the same local network as the PingDirectoryProxy server.

For example, if you specify `prime-index-source:file,ds`, priming is performed from the `global-index` data file created from the previous run of the directory servers. If the entire global index has been primed previously from a startup or reloaded-index directory server's source, the contents of the global index are written to disk periodically with the `file,ds` configuration.

Configuring all indexes at startup

Steps

- To prime all indexes at startup, run the `dsconfig` tool.

  Example:

  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 server's source, if necessary.

```
$ 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 don't want to configure priming during setup, you can configure the index priming manually.

To do this, create an external server, create a global attribute index, and change the entry-balancing request processor to load indexes from the external server.

Steps

1. To create an external server of the PingDirectoryProxy type to represent a peer of the PingDirectoryProxy server, use the `dsconfig` tool with the `create-external-server` subcommand.

   Example:

```
$ 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.

   Example:

   ```
   $ bin/dsconfig create-global-attribute-index \
     --processor-name dc_example+dc+com-eb-req-processor \
     --index-name uid \
   ```

3. To load the indexes at startup from the peer PingDirectoryProxy server, change the entry-balancing request processor using `dsconfig set-request-processor-prop`.

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.

You can configure an Entry Balancing Request Processor to persist the global index to disk periodically 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. Reading the index from disk eliminates the load on backend PingDirectory server instances if many PingDirectoryProxy server instances are starting at the same time.

You can configure an entry-balancing request processor to persist the global index to disk by including `file` as one of the prime index sources with the `prime-index-source` property. The `persist-global-index-frequency` property controls the frequency at which the file is written.

The global index must be primed before it's persisted. It can be primed initially using a peer PingDirectoryProxy server or from a backend PingDirectory server.

When new global attribute indexes are added on a running PingDirectoryProxy, the global index can be primed with those attribute indexes by running the `rebuild-index` tool. The `rebuild-index` tool 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 is 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's using a remote priming source. During server startup, the global index priming works by using each configured `prime-index-source` property in the specified order until it's fully primed to take advantage of what is available locally before contacting one or more remote servers.

Steps

- To prime all indexes at startup from a file, run the `dsconfig` tool.

  Example:

  ```
  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 might become overwhelmed and unresponsive because of their method of streaming data.

About this task

To reduce the impact of priming and reloading on these servers, use the `prime-search-entry-per-second` property and the `--searchEntryPerSecond` property of the `reload-index` command. These properties control the rate at which the PingDirectoryProxy 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 doesn't disconnect, this timeout can be used to force a retry of the operation.

> **Tip**
>
> To find the optimum rate, start low and specify a few thousand search entries per second. Then increase as necessary.

To reduce the impact on server performance:

Steps

- To reduce the impact of priming on servers, 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.
- To specify the amount of time an extended operation response listener can be idle while the global index priming is in progress, use the `index-priming-idle-listener-timeout`.

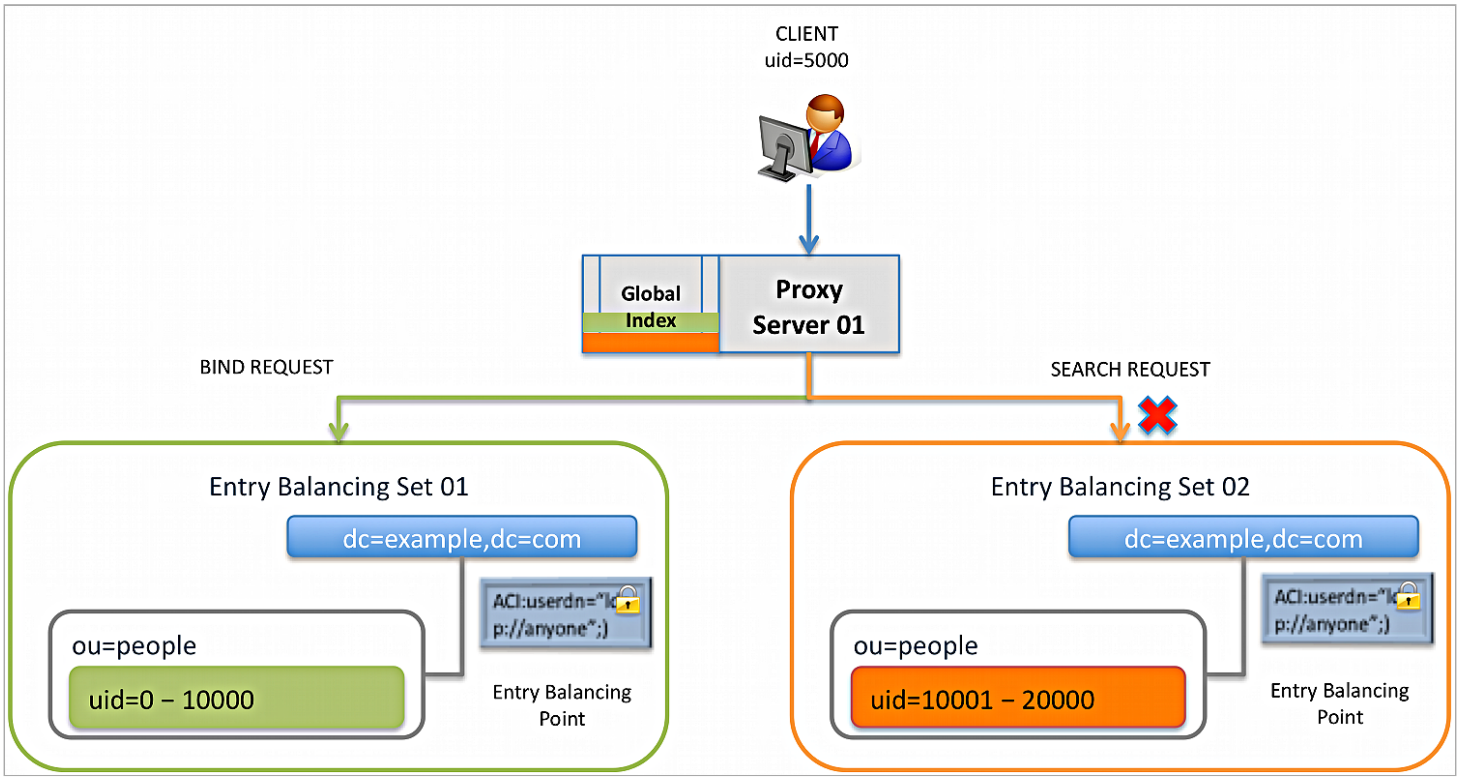
Working with alternate authorization identities

The following sections cover the procedures to configure the alternate authorization identities for the PingDirectoryProxy server.

Access control rules in an entry-balanced deployment are configured in the PingDirectory backend servers and require access to the entry contents of the user issuing the request.

This can introduce a potential issue when clients to the PingDirectoryProxy server authenticate as users whose entries are among the entry-balanced sets. If the server processing a request doesn't contain the issuing user's entry, the access control can't be evaluated. 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 PingDirectory server's in all backend sets and has an equivalent set of access control rights as the authenticated user.

For the following example, assume a deployment has two entry-balancing sets: set-01 and set-02. Set-01 has entries in the `uid=0-10000` range, while set-02 has entries for `uid=10001-20000`.



Entry-Balancing Issue with Clients Not Present in the Underlying Data Set

Processing steps

1. The client with `uid=5000` binds to the PingDirectoryProxy server, which sends a `BIND` request to entry-balancing set-01.
2. The client sends a `SEARCH` request with filter `(uid=15000)`.
3. The PingDirectoryProxy server determines that `uid=15000` lives on entry-balancing set-02.
4. The PingDirectoryProxy server determines that the entry for the authenticated user with `uid=5000` doesn't exist in set-02 and that the access control handler has to reject the `SEARCH` request issued by an unknown user.
5. The PingDirectoryProxy server observes that the PingDirectory server processing a request doesn't contain the entry of the user issuing the request and decides to use an alternate authorization identity.

About alternate authorization identities

Alternate authorization identities allow for the proper evaluation of access control rules for users whose entries aren't present within an entry-balanced dataset.

Whenever the PingDirectoryProxy 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 because the user is known to servers in that set. However, when forwarding a request to a backend set that doesn't contain the user's entry, the PingDirectoryProxy server uses an alternate authorization identity that reflects the generic user with the same set of rights as the actual user issuing the request.

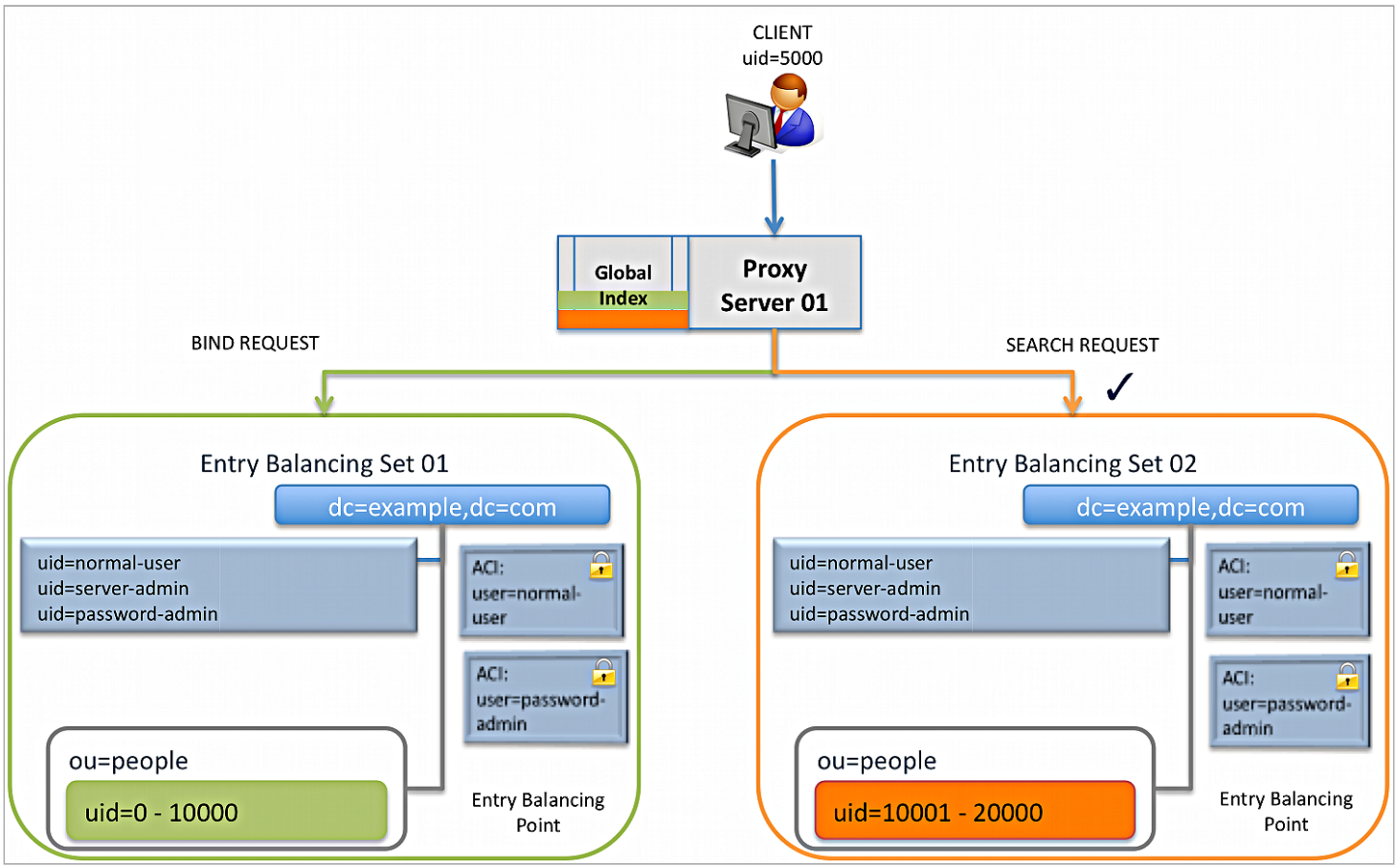
Alternate authorization identities allow for the proper evaluation of access control rules for users whose entries are not present within an entry-balanced dataset.

There are only a few different generic classes of users from an access control perspective. These can be placed in a portion of the directory information tree (DIT) that isn't below the entry-balancing base distinguished name (DN) and is replicated to all servers in the topology.

Whenever a user authenticates to the PingDirectoryProxy 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 determine which of these generic accounts best describes the rights that the user should have.

For the following example, assume that you have three classes of users: full administrators, password administrators, and normal users. Assume that you 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`



Alternate Authorization Identity Solves Access Control Issues in Entry-Balancing Deployments

Processing steps

1. The client with `uid=5000` binds to the PingDirectoryProxy server, which sends a `BIND` request to entry-balancing set-01.
2. The client sends a `SEARCH` request for `uid=15000`.
3. The PingDirectoryProxy server determines that `uid=15000` lives on entry-balancing set-02.

4. The PingDirectoryProxy server then determines that the client `uid=5000` doesn't have an entry on entry-balancing set-02.
5. The PingDirectoryProxy 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` issuing the request.
6. The access control is accepted and the `SEARCH` request returns a response for `uid=5000`.
7. When an alternate authorization identity is invoked, `authzID='dn:uid=normal user,dc=example,dc=com` records in the server log, which indicates the use of the alternate authorization identity.

   For example, if the `user.15000` is in a different backend set from `user.5000`, the log shows the following response.

```
% bin/ldapsearch -D "uid=user.5000,ou=people,dc=example,dc=com" -w pasword \
  -b uid=user15000,ou=people,dc=example,dc=com "(objectclass=)"

[18/Aug/2013:11:54:35 -0500] SEARCH REQUEST conn=153 op=1 msgID=2
via="app='Directory-Proxy address='127.0.0.1'
authzID='dn:uid=normal user,dc=example,dcom' sessionID='conn=2'
requestID='op=1'" base="uid=user.150000,ou=people,dc=example,dc=com"scope=2
filter="(objectclass=)" attrs="ALL"

[18/Aug/2013:11:54:35 -0500] SEARCH REQUEST conn=153 op=1 msgID=2 resultCode=0 etime=2.038
entriesReturned=1 authzDN="uid=normal-user,dc=example,dc=com"
```

Configuring alternate authorization identities

Alternate authorization identities are specified by the `authz-attribute` property of the entry-balancing request processor configuration object.

About this task

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 with a virtual attribute, that value is used to specify the alternate authorization identity for the user. Otherwise, the default authorization identity, as indicated with the `authz-dn` configuration property, is used to determine the alternate authorization identity.

Steps

1. Set the `authz-dn` property of the entry-balancing request processor configuration using the `dsconfig` tool.

   > **Note**
   >
   > If any user among the balanced entries doesn't have an alternate authorization identity defined, the PingDirectoryProxy server uses the value of the `authz-dn` property of the entry-balancing request processor configuration.

   Example:

```
$ 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. Add the following attribute value to a `server-admin` user.

   Example:

   ```
   ds-authz-map-to-dn: uid=server-admin,dc=example,dc=com
   ```

Forwarding authorization identities in requests

By default, the PingDirectoryProxy server tries to ensure that requests it forwards to backend servers get processed with the correct user authorization. To do this, the server relies on a function controlled by the `authorization-method` property, located in the LDAP external server configuration for each of the backend server instances.

Considering proxied authorization scenarios

In environments configured with a PingDirectoryProxy server in front of PingDirectory server instances, the `authorization-method` property is typically set to `intermediate-client-control`. This `authorization-method` value might not be the best option when clients need to authorize a request as a user whose account doesn't exist in the backend server, including the following scenarios:

- Entry-balanced configurations where a user whose account resides in one backend set might need to issue requests targeting entries in a different backend set
- Configurations that have multiple subtree views backed by different sets of backend servers for different parts of the DIT, where a user whose account resides in one part might need to issue requests targeting entries in a different part

However, the `intermediate-client-control` setting can still be appropriate for deployments where some users might need to issue requests that get processed by servers that don't contain their accounts. In these scenarios, there are a small number of roles that any user can assume, such as a regular end user, a password administrator, or a full server administrator. You can create surrogate accounts for each of those roles that reside in all the backend servers.

> **Tip**
>
> To indicate that operations processed in servers that don't contain the requester's user entry should be authorized as the appropriate surrogate account, use the `ds-authz-map-to-dn` operational attribute, whether real or virtual.

Using the `forward-authorization-entry-control`

You can also use the `forward-authorization-entry-control`, which causes the PingDirectoryProxy server to forward a copy of the requester's entry to the backend server. The PingDirectoryProxy server uses that entry to authorize requests as that user in backend servers that don't already contain the entry.

Important

You should prefer the `intermediate-client-control` for proxied authorization.
Reserve the `forward-authorization-entry-control` for scenarios where it's common for users in one entry-balanced backend set to need access to entries in other backend sets—but whose entries can't be reasonably mapped to surrogate entries.
You can only use the `forward-authorization-entry-control` in topologies where all servers are running version 10.3 or later.

Steps

For each PingDirectoryProxy server in the topology, make the following configuration change for every LDAP external server instance:

- **Set the value of `authorization-method` to `forward-authorization-entry-control`.**

  Example:

```
$ bin/dsconfig set-external-server-prop \
  --server-name server.example.com:636 \
  --set authorization-method:forward-authorization-entry-control
```

Managing entry-balancing replication

Replication in the PingDirectoryProxy server synchronizes directory data between all servers in the topology.

However, In a deployment using the entry-balancing feature, 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 section about replication in an entry-balancing environment contains the following:

- Overview of replication in an entry-balancing environment
- Replication prerequisites in an entry-balancing deployment
- About the --restricted argument of the dsreplication command-line Tool
- Checking the status of replication in an entry-balancing deployment
- Example of configuring entry-balancing replication

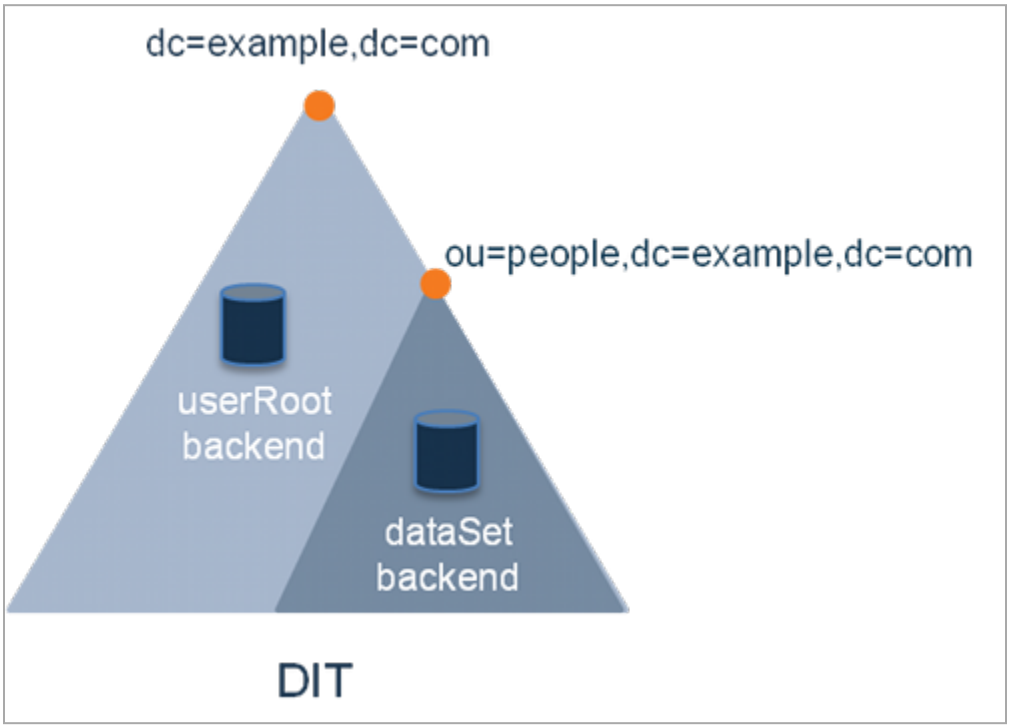
Overview of replication in an entry-balancing environment

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 distinguished name (DN) and must be a base DN of a backend.

By default, replication propagates updates to all replication servers in the topology. However, updates to data under the entry-balancing point must be replicated only among server instances in the same data set. Replication requires that in these deployments the PingDirectory 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`.

The following illustration demonstrates the global and restricted backends in a directory information tree (DIT).



If a data set name is not defined when you set up the PingDirectoryProxy server, one is provided by default. The proper configuration of an entry-balancing environment requires coordination between the PingDirectory server and the PingDirectoryProxy server. After you enable replication, you can designate the replication domain as the domain participating in entry balancing.

You can find more details about replication, managing the replication topology, and working with multiple backends, in the PingDirectory Server Administration Guide.

Replication prerequisites in an entry-balancing deployment

Before you can enable replication in an entry-balancing deployment, you must have the following set up in your server topology:

Multiple local DB backends

When you set up the PingDirectory server instances, you need two backends:

- A global backend for globally replicated data, such as `userRoot`
- A backend for the balancing point base distinguished name (DN), such as `dataSet`

Both backends must be enabled for replication and initialized separately.

Replication set name

Every PingDirectory server in your replicated topology must have a replication set name. This replication set name coordinates the PingDirectoryProxy server and the PingDirectory server.

The restricted domain is only replicated within instances using the same replication set name.

Multiple PingDirectoryProxy 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 has a proxying request processor associated with it. The restricted base DN has an entry-balancing request processor associated with it.

> **Tip**
>
> Configure the subtree views after running your setup using the `create-initial-proxy-config` tool.

Homogeneous domains

All servers in a replication topology must have the same domains.

> **Note**
>
> Because the PingDirectory server replication topology assumes that all domains are on all servers, it is not suitable for environments with an inconsistent presence of domains across the server topology.

About the --restricted argument of the dsreplication command-line Tool

The `dsreplication` tool with the `--restricted` argument specifies a base distinguished name (DN) when multiple domains are enabled in an entry balanced environment.

When enabling replication for a server that takes part in an entry-balanced environment, the multiple domains involved are enabled at the same time. There is a global domain and a restricted domain. The restricted domain represents the entry-balancing point. Each base DN is defined in a separate local database (DB) backend. Use the `dsreplication` command-line interface (CLI) tool with the `--restricted` argument to specify which base DN is considered an entry-balancing point.

Using the --restricted argument of the dsreplication command-line tool

About this task

To enable replication between two servers with entry balancing:

Steps

- Choose one of the following ways to run the `dsreplication` command:

  Choose from:

  - Run `dsreplication` in non-interactive mode.

    See the following example.

```
$ 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
```

- Run `dsreplication` using the interactive command line.

  1. Enter `dsreplication` and the LDAP connection parameters.
  2. Specify that an entry balancing is being used.
  3. Specify the base distinguished name (DN) of the entry-balancing point.

     See the following example.

```
You must choose at least one base DN to be replicated.

Replicate base DN dc=example,dc=com? (yes / no) [yes]: yes

Replicate base DN ou=people,dc=example,dc=com? (yes / no) [yes]: yes

Do you plan to configure entry balancing using the Directory Proxy Server? (yes / no)
[no]: yes

Is dc=example,dc=com an entry-balancing point? (yes / no) [no]: no

Is ou=people,dc=example,dc=com an entry-balancing point? (yes / no) [no]: yes
```

Checking the status of replication in an entry-balancing deployment

You can use the `dsreplication` status tool to check the status of an entry-balancing deployment.

About this task

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

- To get a status of replication in the entry-balancing deployment, run the `dsreplication` command.

  Example:

> **Note**
>
> By default, all of the sets are displayed. To view only a specific replication set, use the `--setName` option.

Example:

```
$ bin/dsreplication status --hostname host1.example.com \
  --port 1389 --adminUID admin --adminPassword secret
```

Result:

```
          --- 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 covers how to set up a four-server replication topology that uses entry balancing to distribute entries across the servers and provides a start-to-finish example to walk you through this process.

Assumptions

The example assumes the following conditions:

- None of the servers have participated in any previous replication topology. This is supported for one or multiple entry balancing domains.
- This example uses the LDAP (389) and replication (8989) ports. It configures the following hosts:
  - austin1.example.com
  - newyork1.example.com
  - austin2.example.com

- newyork2.example.com

- The global domain is `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 or New York.

Configuration summary

To configure replication in an entry-balanced deployment:

1. Install two directory servers in an Austin location and two in a New York location.
2. Create a new backend, called `dataset`, to store the entry-balancing data set.
3. Define entry-balancing set names `dataSet1` and `dataSet2` for assignment to the `replication-set-name` Global Configuration Property of the PingDirectory server instances.
4. Import the data representing the global domain, stored in `userRoot`, into a server. Choose a server for each of the entry-balancing data sets, which are both stored in the backend named `dataset`.
5. Enable replication and initialize remaining servers.
6. Configure the proxies.
7. Check the status of replication.

Installing the PingDirectory server
About this task

Install four PingDirectory server instances, two in the Austin location and two in the New York location:

- austin1.example.com
- newyork1.example.com
- austin2.example.com
- newyork2.example.com

Steps

1. Install the first Austin server, `austin1`.

   Example:

   ```
   root@austin1# ./setup --baseDN dc=example,dc=com \
   --ldapPort 389 --rootUserDN "cn=Directory Manager" \
   --rootUserPassword pass --no-prompt --acceptLicense \
   --instanceName ds1 --location Austin
   ```

2. Install the second Austin server, `austin2`.

   Example:

```
root@austin2 # ./setup --baseDN dc=example,dc=com \
--ldapPort 389 --rootUserDN "cn=Directory Manager" \
--rootUserPassword pass --no-prompt --acceptLicense \
--instanceName ds2 --location Austin
```

3. Install the first New York server, `newyork1`.

   Example:

   ```
   root@newyork1# ./setup --baseDN dc=example,dc=com \
   --ldapPort 389 --rootUserDN "cn=Directory Manager" \
   --rootUserPassword pass --no-prompt --acceptLicense \
   --instanceName ds3 --location NewYork
   ```

4. Install the second New York server, `newyork2`.

   Example:

   ```
   root@newyork# ./setup --baseDN dc=example,dc=com \
   --ldapPort 389 --rootUserDN "cn=Directory Manager" \
   --rootUserPassword pass --no-prompt --acceptLicense \
   --instanceName ds4 --location NewYork
   ```

Creating the database backends and defining the replication set name
About this task

To store the entry-balancing data set, create a new backend.

Steps

1. On each server, create a dataset backend named `dataset`.

   Example:

   ```
   ./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`.

   Example:

   ```
   ./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`.

   Example:

```
./bin/dsconfig --no-prompt \
set-global-configuration-prop \
--set replication-set-name:dataset2
```

Creating and setting the locations
About this task

Create and set the locations of the Austin and New York server instances.

Steps

1. On the Austin servers, create two instance locations, `newyork` and `austin`.

   Example:

   ```
   ./bin/dsconfig --no-prompt create-location --location-name austin

   ./bin/dsconfig --no-prompt create-location --location-name newyork \
   --set preferred-failover-location:austin
   ```

2. Set the location of the Austin server instances to `austin`.

   Example:

   ```
   ./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
   ```

3. On the New York servers, create two instance locations, `newyork` and `austin`.

   Example:

   ```
   ./bin/dsconfig --no-prompt create-location \
   --location-name austin

   ./bin/dsconfig --no-prompt create-location \
   --location-name newyork \
   --set preferred-failover-location:austin
   ```

4. Set the location of the New York server instances to `newyork`.

   Example:

   ```
   ./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

Import the `userRoot` data based on data defined in the `userRoot.ldif` file into one server.

About this task

Choose a server for each of the entry-balancing data sets, both stored in the dataset backend.

> **Note**
>
> The `userRoot.ldif` file doesn't contain entries at or within the entry-balancing point, `ou=people,dc=example,dc=com`.

Steps

1. **Import the userRoot data using the `import-ldif` command.**

   Example:

   ```
   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 that is assigned the `replication-set-name` on one server into the dataset backend.**

   Example:

   ```
   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 that is assigned the `replication-set-name` on one server into the dataset backend.**

   Example:

   ```
   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

Enable replication between the servers and initialize the remaining servers without data.

About this task

> **Note**
>
> In this example, the `--restricted` domain is specified in the **`dsreplication`** command.

To enable replication on the servers in the topology:

Steps

1. Run `dsreplication enable`.

   Example:

   ```
   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
   ```

   > **Note**
   >
   > Running the **`dsreplication enable`** command for the first time creates the administrator account.

2. Enable replication between `austin1` and `newyork1`.

   Example:

   This procedure automatically also enables replication between `austin2` and `newyork1`.

   ```
   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`.

   Example:

```
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
```

Result:

The entry-balancing replication setup is complete.

4. Initialize the remaining servers without data.

    1. Initialize the global domain, `dc=example,dc=com` on `austin2`, `newyork1`, and `newyork2` servers.

        Example:

        ```
        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
        ```

    2. Initialize the entry-balancing domain, `ou=people,dc=example,dc=com`, from `austin1` to `newyork2`.

        Example:

        ```
        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
        ```

    3. Initialize the entry-balancing domain, `ou=people,dc=example,dc=com`, from `austin2` to `newyork2`.

        Example:

```
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
```

> **Note**
>
> After `austin2` is initialized with the global domain, you can combine steps 4b and 4c by initializing both domains with one invocation.

Checking the status of replication

After you have configured replication, check the status of the replication topology using the `dsreplication status` command.

Steps

- To check the replication topology status, run the `dsreplication status` command.

  Example:

  ```
  root@austin1# ./bin/dsreplication status \
  --adminPassword pass --no-prompt --port 389
  ```

Managing the PingDirectoryProxy server

After you have configured the PingDirectoryProxy server, you can manage the day-to-day operations of your deployment using the monitoring and logging features.

Managing logs

The PingDirectoryProxy server provides several different log publisher types that provide information about how the server is processing.

About the default logs

You can view all logs in the `PingDirectoryProxy/logs` directory. The following logs are documented in detail:

- 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. Several 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, and 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, because the server provides a separate debug logging facility.
- 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 can be filtered based on their severities in two ways. First, the error log publisher has a `default-severity` property, which can 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 can be altered on the fly using tools like `dsconfig`, the admin 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 can be activated with a different set of default severities to debug a short-term problem, and then that logger can be removed after 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 several 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 could occur in the course of processing.

The audit 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 reduces the server's write performance.

By default, if you enable the audit log on the 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 can 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 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 can 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 can 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 can contain.

The 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 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` command 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 might be used if a file with that name already exists because the `setup` command has already been run. The full path to the setup log file is provided when the `setup` command has completed.

Tool log

Many of the administrative tools provided with the 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 using the server tasks interface, log messages generated while the task is running can 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 PingDirectory server and the PingDirectoryProxy 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 different types of loggers that 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, debugging problems with directory-enabled applications, and 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

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 following log publishers are available for the PingDirectoryProxy server but 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 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.

   Example:

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 `--set compression-mechanism: gzip` option to the previous command.
Because compression can't be disabled or turned off after being configured for the logger, plan carefully to determine your logging requirements, including log rotation and retention with regard to compressed logs.

2. View log publishers with the following command.

```
$ bin/dsconfig list-log-publishers
```

Creating a log publisher using dsconfig interactive command-line mode

Steps

1. In the command line, enter `bin/dsconfig`.
2. Authenticate to the server by following the prompts.
3. In the main menu, select the option to configure the log publisher.
4. In the `Log Publisher menu`, select the option to create a new log publisher.
5. Select the Log Publisher type.

   For this example, select File-Based Access Log Publisher.
6. Enter a name for the log publisher.
7. Enable it.
8. Enter the path to the log file relative to the PingDirectory server root.

   For this 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. In the `Log Publisher Properties menu`, select the option for `log-connects:false`, `log-disconnects:true`, `log-requests:false`, and `log-results:false`.

12. Enter `f` to apply the changes.

About log compression

The 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 when the logger is configured. Because of problems in trying to append to an existing compressed file, if the server encounters an existing log file at startup, it rotates 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 might want to have a second logger defined that does not use compression and has rotation and retention policies that minimizes 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.

Configure compression by setting the `compression-mechanism` property to have the value of `gzip` when creating a new logger.

About log signing

The 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 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.

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 is not completely verifiable because it still contains unsigned content from before signing was enabled. Only log files whose entire content was written with signing enabled are considered completely valid. For the same reason, if a log file is still open for writing, then signature validation does not indicate that the log is completely valid because the log doesn't 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.

After 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 server lets you encrypt log files as they are written.

The `encrypt-log` configuration property controls whether encryption is enabled for the logger. Enabling encryption causes the log file to have an `.encrypted` extension. If both encryption and compression are enabled, the extension is `.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 can't be turned on or off after 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 reduces 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 is performed with the server's preferred encryption settings definition.

To explicitly specify which definition should be used for the encryption, set the `encryption-settings-definition-id` property with the ID of that definition. You should set the encryption settings definition to be 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. You can also create a randomly generated encryption settings definition, 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 might remain in an in-memory buffer until the log file is closed. The encryption is performed using a block cipher, and it can't write an incomplete block of data until the file is closed. This is not an issue for any log file that isn't 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

Configure log signing for a log publisher.

Steps

1. To enable log signing for a log publisher, use `dsconfig`.

   Example:

   In this example, the `sign-log` property is set 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.

   Example:

   ```
   $ 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

The server provides a tool, `validate-file-signature`, that checks if a file hasn't been tampered with in any way.

Steps

- Run the `validate-file-signature` tool to check if a signed file has been tampered with.

  Example:

  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
  ```

  Result:

  ```
  All signature information in file 'logs/audit' is valid
  ```

  > **Note**
  >
  > If any validation errors occur, you will see a message similar to the following:
  >
  > ```
  > 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

Configure log file encryption for a log publisher.

Steps

1. To enable encryption for a log publisher, use `dsconfig`.

   Example:

   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. Decrypt and decompress the file.

   Example:

```
$ 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/Data-Sync/decrypted-access' using a
key generated from encryption settings definition
'332c846ef0dcd1d5187c1592e4c74cad33fc1e5fc20b726cd301cdd2b3ffbc2b'
Successfully wrote 123,456,789 bytes of decrypted data
```

Configuring log rotation

The PingDirectory server allows you to configure the log rotation policy for the server.

About this task

When any rotation limit is reached, the 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.

  Example:

```
$ 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 server provides two log file rotation listener, the copy log file rotation listener and the summarize log file rotation listener, which you can enable with a log publisher.

About this task

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 is subject to deletion by a log file retention policy, but the copy is not automatically removed.

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.

Steps

- 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
```

> **Note**
>
> 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 are 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.

- **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
```

> **Note**
>
> 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

The server allows you to configure the log retention policy for each log on the server.

About this task

When a retention limit is reached, the server removes the oldest archived log before 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 server to retain. The default file count is 10 logs. If the file count is set to 1, then the log continues 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 server's requirements. This requires 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.

  Example:

```
$ 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 PingDirectoryProxy 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 is always honored. For example, if the server-wide maximum concurrent connections limit is reached, then all subsequent connections are 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:

`maximum-concurrent-connections`

Specify the maximum number of client connections that can be established at any given time.

If the server already has the maximum number of connections established, then any new connection attempts from any clients are rejected until an existing connection is closed.

The default value of zero indicates that no limit is enforced.

`maximum-concurrent-connections-per-ip-address`

Specify the maximum number of client connections that can be established at any given time from the same client system.

If the server already has the maximum number of connections established from a given client, then any new connection attempts from that client are rejected until an existing connection from that client is closed. The server continues to accept connections from other clients that haven't yet reached this limit.

The default value of zero indicates that no limit is enforced.

`maximum-concurrent-connections-per-bind-dn`

Specify the maximum number of client connections that can be established at any given time while authenticated as a particular user.

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 a given user reaches the maximum number of connections, then any new attempt to authenticate as that user causes the connection performing the bind to be terminated.

> **Note**
>
> This limit applies only to authenticated connections and isn't enforced for clients that haven't 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 take effect only for new connections established after the change is made. Any change to the `maximum-concurrent-connections-per-bind-dn` property applies only to connections, including existing connections, that perform authentication after the change is made.

Existing connections are allowed to remain established even if that would cause the new limit to be exceeded.

Setting client connection policy resource limits

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 can be associated with a specific client connection policy at any given time. After this limit has been reached, any further attempts to associate a connection with this client connection policy 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 can persist. After this period, the connection is terminated.

`maximum-idle-connection-duration`

This property specifies the maximum time that a connection associated with a particular client connection policy can persist 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 are terminated.

`maximum-operation-count-per-connection`

This property specifies the maximum number of operations that can be requested by any client connection associated with this client connection policy. Attempts to process more than this number of operations on the connection terminates the connection.

`maximum-concurrent-operations-per-connection`

This property specifies the maximum number of concurrent operations 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 can issue requests to the PingDirectoryProxy server. If a client attempts to request operations at a rate higher than this limit, then the server behaves 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 can issue requests to the PingDirectoryProxy server. If this limit is exceeded, then the server responds as specified in the `policy-operation-rate-exceeded-behavior` property.

`policy-operation-rate-exceeded-behavior`

This property specifies the behavior of the PingDirectoryProxy server if a client connection attempts to exceed the rate defined in the `maximum-policy-operation-rate` property.

Monitoring the PingDirectoryProxy server

While the PingDirectoryProxy 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 Java Management Extensions (JMX)

Monitoring the server using the status tool

The PingDirectoryProxy server provides a `status` tool that provides basic server status information, including version, connection handlers, a table of LDAP external servers, and the percent of the global index being used.

Steps

1. To view the current state of the server, run the `status` tool.

```
$ 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':
```

Result:

```
          --- 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 8.0.0.0
Java Version:         jdk-7u9

          --- Connection Handlers ---

Address:Port : Protocol : State
-------------:----------:---------

Step 0.0.0.0:1689 : JMX      : Disabled
Step 0.0.0.0:636  : LDAPS    : Disabled
Step 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
```

Monitor entries

> **Note**
>
> This topic applies only to the PingDirectory server.

While the 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's 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 environment used by the associated backend.
- Disk Space Usage Monitor Entry (cn=Disk Space Usage,cn=monitor) – Provides information 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 percentage of operations that were 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 server version.
- Work Queue Monitor Entry (cn=Work Queue,cn=monitor) – Provides information about the state of the server's 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

Alarms, alerts, and gauges alert administrators to changes in server conditions that might require attention.

Alarms

An alarm represents a stateful condition of the server or a resource that might indicate a problem, such as low disk space or external server unavailability.

Alarms have severity, name, and message. Alarms will always have a Condition property, and can have a Specific Problem or Resource property. If surfaced through SNMP, a Probable Cause property and Alarm Type property are also listed. You can configure alarms to generate alerts when the alarm's severity changes.

You can configure the Alarm Manager, which governs the actions performed when an alarm state is entered, through the `dsconfig` tool and the admin console. A complete list of system alerts, alarms, and their severity is available in `<server-root>/docs/admin-alerts-list.csv`.

The server complies with the International Telecommunication Union CCITT Recommendation X.733 (1992) standard for generating and clearing alarms. If configured, entering or exiting an alarm state might 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. You can view alarms with the `status` tool. As with other alert types, you can configure alert handlers can to manage the alerts generated by alarms.

Alerts

There are two alert types supported by the server: standard and alarm-specific.

The server constantly monitors for conditions that might need administrator attention, such as low disk space. For this condition, the standard alert is `low-disk-space-warning`, and the alarm-specific alert is `alarm-warning`.

You can configure the server to generate alarm-specific alerts as well as standard alerts. By default, standard alerts are generated for conditions internally monitored by the server. However, gauges can only generate alarm-alerts.

Gauges

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.

Numeric gauges monitor continuous values like CPU load or free disk space. Indicator gauges monitor enumerated set of values such as `server available` or `server unavailable`. Gauges generate alarms when the gauge's severity changes because of changes in the monitored value.

The server installs a set of gauges that are specific to the product and that can be cloned or configured through the `dsconfig` tool. You can tailor existing gauges to fit each environment by adjusting the update interval and threshold values. Configuration of system gauges determines the criteria by which alarms are triggered.

Use the Stats Logger to view historical information about the value and severity of all system gauges. For more information, see Profiling server performance using the Stats Logger.

Testing alarms and alerts

Steps

1. Use `dsconfig` to configure a gauge and set the `override-severity` property to critical.

   The following example configures the CPU Usage (Percent) gauge.

   Example:

   ```
   $ 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.

   Example:

   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

The server raises indeterminate alarms for a server condition for which a severity can't be determined.

In most cases, these alarms are benign and don't issue alerts, nor do they appear in the output of the `status` tool or admin console by default.

These alarms are usually caused by an enabled gauge intended to measure an aspect of the server that isn't currently enabled. For example, gauges intended to monitor metrics related to replication might produce indeterminate alarms if a server isn't currently replicating data. The gauge can be disabled if needed.

For more information about indeterminate alarms, view the gauge's associated monitor entry. There might 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. A search of the monitor backend for this gauge's entry results in an error message that might 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
              …
```

Administrative alert handlers

The 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.

> **Important**
>
> SNMP is deprecated.

The PingDirectory provides several 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 PingDirectory. Information about the administrative alert will be made available to the executed application as arguments provided by the command. Learn more in Changing the timeout for an exec alert handler.

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 primary 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 Java Management Extensions (JMX) connection handler and the alert handler using the `dsconfig` tool.

To allow users to receive JMX notifications, enable the `jmx-read` and `jmx-notify` privileges. By default, these privileges aren't granted to any users, including root users or global administrators.

For security reasons, you should create a separate user account that has only these privileges.

You can also configure the JMX connection handler and the alert handler using `dsconfig` in interactive command-line mode, which is visible in the `Standard object menu`.

Configuring the JMX connection handler

About this task

To enable the JMX connection handler:

Steps

1. Run `dsconfig`.

   Example:

   ```
   $ 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 using the `ldapmodify` tool using an LDIF representation.

   Example:

   ```
   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

About this task

To configure the Java Management Extensions (JMX) alert handler:

Steps

- Run `dsconfig`.

  Example:

  ```
  $ bin/dsconfig set-alert-handler-prop --handler-name "JMX Alert Handler" \
    --set enabled:true
  ```

Configuring the SMTP alert handler

To create a new instance of an SMTP alert handler, use the `dsconfig` tool.

About this task

By default, there is no configuration entry for an SMTP alert handler.

Steps

- Use the `dsconfig` tool to configure the SMTP Alert Handler.

  Example:

```
$ 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 in the `Standard object menu`.

About this task

Before you begin, you need an SNMP subagent capable of communicating using SNMP2c. For more information about SNMP, see Monitoring using SNMP.

The PingDirectory server also supports an SNMP alert handler, which is used in deployments that do not enable an SNMP subagent.

To configure the SNMP subagent alert handler:

Steps

- Run `dsconfig`.

  Example:

  In this example:

  - 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.

```
$ 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 PingDirectoryProxy server data.

The proxy virtual attributes apply to a local proxy backend, such as `cn=config` or the Root DSA-specific entry (DSE).

If you want virtual attributes in entries for proxied requests, they must be configured in the backend servers. Alternately, attributes can be inserted into those entries using proxy transformations. For more information, see Configuring proxy transformations.

For example, you can define a virtual attribute and assign it to the Root DSE using `create-virtual-sttribute`.

```
$ 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 `Primary Proxy`.

```
$ bin/ldapsearch --baseDN "" --searchScope base --bindDN "" \
  --bindPassword "" --port 5389 -- hostname localhost \
  "objectclass=*" description

dn:
description:PrimaryProxy
```

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-primary 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, PingDirectory 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. Before 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.

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 PingDirectory servers as pets today can take advantage of current DevOps and IaC principles to turn them into cattle.

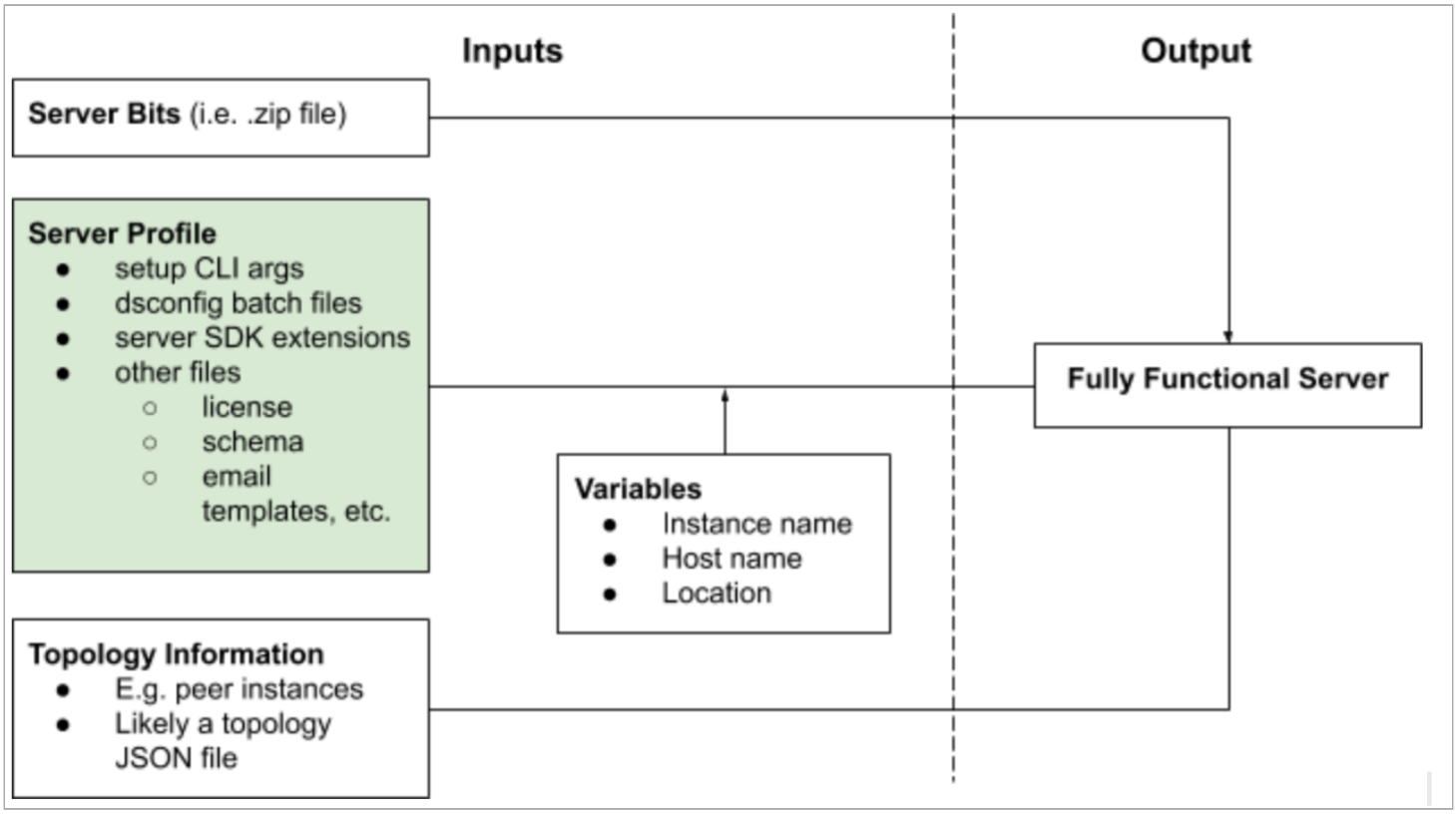
Server profiles

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 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.
- 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 before 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.

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 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/
  |-- pre-setup-dsconfig/
  |-- 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 `setup-arguments.txt` file, located in the `resources/server-profile-template.zip` archive.

When you run the `manage-profile setup` command, these arguments are passed to the server's setup tool. To view the arguments available in this file, run the server's `setup --help` command.

To provide the equivalent, non-interactive command-line interface (CLI) arguments after any prompts have been completed, run `setup` interactively. The `setup-arguments.txt` file in the server profile template contains an example set of arguments that can be changed.

`setup-arguments.txt` is the only required file in the profile.

dsconfig/ and pre-setup-dsconfig/

You can include `dsconfig` batch files in the server profile with information about changes to apply to the out-of-box system configuration. Any batch files in the server profile must include a `.dsconfig` extension. `dsconfig` batch files in the server profile can't make changes to any topology configuration, including topology admins and server instances, so you should make topology configuration changes outside of the server profile.

The batch files can be placed in the following server profile directories:

`dsconfig/`

Contains information about changes to apply after the `manage-profile` tool runs `setup`, copies any post-setup files into place, and installs any Server SDK extensions contained in the server profile. The `dsconfig/` directory should be used for most configuration changes applied to the server.

`pre-setup-dsconfig/`

Holds information about changes to apply immediately before the `manage-profile` tool runs `setup`. You should only use the `pre-setup-dsconfig/` directory when setting up the server with a pre-existing encryption settings database and applying the changes required for configuring and activating the cipher stream provider needed to access the encryption settings database.

Note

If multiple batch files are present in the directory that the `manage-profile` tool is using, the files are processed in ascending lexicographic order. For example, `00-base.dsconfig` is processed before `01-second.dsconfig`.

Tip

You can use the `config-diff` tool to create a `dsconfig` batch file that reproduces the current topology configuration.

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 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.

The `--excludeSetupArguments` option generates a server profile without a setup-arguments.txt file. This is useful when generating server profiles for use with Docker images.

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
- 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.

The `--skipValidation` option which skips the validation step when running on an offline server

> **Important**
>
> When you run the **`manage-profile replace-profile`** tool with an SDK extension included in the new profile, the tool invokes the **`setup`** command.
> The **`manage-profile replace-profile`** tool can update the server version when needed, but will fail if you attempt to downgrade the server to an earlier version. It can also 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.

Managing Server SDK extensions

The 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 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 can 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 several different types of extensions for Ping Identity server products, including the PingDirectory server, the PingDirectoryProxy server, and the 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 server provides a full suite of command-line tools necessary for administration. 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

PingDirectory provides the following command-line tools, which you can run in interactive, non-interactive, 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`

Command-Line Tools

`audit-data-security`	Invoke data security audit processing in order to identify potential risks or other notable security characteristics contained in directory data. This tool schedules an internal task with the server 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. The list of available auditors can be obtained using `dsconfig list-data-security-auditors --advanced --property name`. Either the `--includeAuditor` or the `--excludeAuditor` arguments can be used to limit the scope of the audit. Additionally, the entries scanned can be limited by specifying the backends to scan, or by specifying an LDAP filter that is used to selected entries to be processed. This tool schedules an operation to run within the PingDirectory server's process. LDAP connection options must be supplied 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.
`authrate`	Perform repeated authentications against the PingDirectory server, where each authentication consists of a search to find a user followed by a bind to verify the credentials for that user.
`backup`	Back up one or more directory server backends. Each backend backup is stored in a separate backend backup directory. A backend backup directory can contain multiple backups of the backend. Each backend backup directory contains a `backup.info` file providing information about each backup in the directory and an archive file for each backup. The name of the archive file includes both the backend ID and the backup ID. The backup ID can be provided to the backup command, or an ID is generated from a current timestamp. Each backup can be optionally compressed, encrypted, hashed or signed. A backup taken on one system can be restored on another system. This tool features both an offline mode of operation as well as the ability to schedule an operation to run within the directory 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.
`base64`	Encode raw data using the base64 algorithm or decode base64-encoded data back to its raw representation.
`check-replication-domains`	Scan the `replicationChanges` database for all known replication domains and identify any obsolete replicas still listed as part of a topology. Run this tool before upgrading replicated topologies to help avoid lockdown mode. Learn more about Discovering obsolete replicas.

`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. Information collected can include configuration files, server monitor entries, portions of log files, JVM thread stack dumps, system metrics, and other data that might be helpful in diagnosing problems, understanding server performance, or otherwise assisting with support requests. Although the tool will do its best to obscure or omit sensitive data, and the entire archive can be encrypted if you desire, you might want to review the resulting support data archive to ensure verify its contents. Further, the archive will include a summary of any potential problems or concerns that are identified in the course of collecting the support data.
`compare-ldap-schemas`	Compares the schemas of two LDAP servers to identify schema elements that might be present in one but not the other, or elements that might be present in both servers but have differences between them.
`config-diff`	Compares directory server configurations and produces a `dsconfig` batch file needed to bring the source inline with the target. Its uses include comparing multiple servers for configuration differences, producing a batch file to reconfigure a server from scratch from the out-of-the-box configuration, and comparing a local server against an expected configuration. Both the source and the target configurations can be retrieved over LDAP, accessed from the local server's file system, extracted from a specific file, or retrieved from every server in a configuration server group. Also, with the exception of accessing a configuration from a specific file, the source and/or target configurations can be compared as they existed at any point in the past, including the baseline, pre-installation configuration. Some configuration differences (those that will always differ between instances, like instance-name) are excluded by default to reduce the amount of spurious output, but these can be included by specifying the `--includeExpectedDifferences` command. Further differences can be excluded with the `--exclude` option. This tool attempts to generate a batch file that can be applied to the source server without any errors. However, there are some edge case configurations that the tool is not sophisticated enough to handle. For example, it cannot handle two peer configuration objects that would require swapping values for a property (for example, evaluation-order-index) that must be unique within the server. It will still generate a `dsconfig` batch file that includes these changes, but they might be rejected by the server. In these rare cases, the batch file can be hand edited so that it can be applied to a running server or it can be applied with the server shut down using `dsconfig --offline`.
`create-rc-script`	Create a Run Control (RC) script that you can use to start, stop, and restart the PingDirectory server on UNIX-based systems.
`create-systemd-script`	Create a systemd script to start and stop the PingDirectory server on Linux-based systems.

`dbtest`	Inspect the contents of the PingDirectory 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. Each local DB backend has a root container, identified by the backend ID. Each root container has an entry container for each base distinguished name (DN) in the backend. Each entry container has several database containers that store entries, attribute indexes and system indexes. The `dbtest` command allows the contents of root containers, entry containers and database containers to be inspected.
`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 PingDirectory server configuration. This utility offers three primary modes of operation, the interactive mode, the non-interactive mode, and the batch mode: • The interactive mode supports viewing and editing the configuration using an intuitive, menu driven environment. Running `dsconfig` in interactive provides a user-friendly, menu-driven interface for accessing and configuring the server. To start `dsconfig` in interactive command-line mode, simply invoke the `dsconfig` shell script or batch file without any arguments. • The non-interactive mode provides a simple way to make arbitrary changes to the PingDirectory server by invoking it on the command-line. If you want to use administrative scripts to automate the configuration process, then run the `dsconfig` command in non-interactive mode. • The batch mode reads multiple `dsconfig` invocations from a file and executes them sequentially. The batch file provides advantages over standard scripting in that it minimizes LDAP connections and JVM invocations that normally occur with each `dsconfig` call. You can view the `logs/config-audit.log` file to review the configuration changes made to the PingDirectory server and to use them in the batch file.

`dsjavaproperties`	Configure the JVM options used to run the PingDirectory server and its associated tools. The options managed by this tool are stored in `config/java.properties`. Typically, you should not edit that file directly. Instead, run the tool specifying `--jvmTuningParameter` arguments to customize JVM options appropriate for this system. Note that the changes will only apply to this PingDirectory server installation. No modifications will be made to your environment variables. Memory and other settings for the JVM tools, including the **`start-server`** tool, can be tuned during initialization by specifying one or more instances of the `--jvmTuningParameter` option when invoking this tool. Supported values are as follows: • `NONE`: Explicitly specify no parameters. • `AGGRESSIVE`: This system is dedicated to running only this server. The amount of memory allocated to this server will be computed accordingly. • `SEMI_AGGRESSIVE`: This system is shared by multiple server processes. The amount of memory allocated to this server will be computed accordingly. If no parameters are specified, the parameters specified by the previous invocation of this tool or setup will be used. Use the `NONE` option to explicitly specify no parameters.
`dsreplication`	Manage data replication between two or more PingDirectory server instances. For replication to work, you must first to enable replication using the **`enable`** subcommand. Then, you initialize the contents of one of the servers with the contents of the other using the **`initialize`** subcommand.
`dump-dns`	Obtain a listing of all of the DNs for all entries below a specified base DN in the PingDirectory 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, LDAP Data Interchange Format (LDIF) exports, and log files generated by the server.
`encryption-settings`	Manage the server encryption settings database. More information about the cipher algorithms and transformations available for use can be found in the Java Cryptography Architecture Reference Guide, as well as the Standard Algorithm Name Documentation for your chosen JDK implementation used by this server.

`enter-lockdown-mode`	Request that the PingDirectory server enter lockdown mode, during which it only processes operations requested by users holding the `lockdown-mode` privilege. While in lockdown mode, the PingDirectory server rejects all requests from users that do not hold the `lockdown-mode` privilege.
`export-ldif`	Export data from the PingDirectory server backend in LDIF form. To export data from a remote PingDirectory server, the server must be running and connection parameters must be supplied. You can specify options to include or exclude specific attributes and branches of the tree, and to include or exclude entries matching a given filter. The data can be appended to an existing file instead of overwriting it, and the output can be optionally compressed. This tool features both an offline mode of operation as well as the ability to schedule an operation to run within the PingDirectory 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.
`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. Include the following arguments to configure the output of the tool: **`--filter`** The export only includes entries matching the specified filter. **`--exportNonReversiblePasswords`** The export includes entries with non-reversible passwords. **`--exportEntriesWithoutPasswords`** The export includes entries that don't have passwords. **`--includeAdditionalAttribute`** The exported entries include the specified attribute(s) in addition to the user's DN and password. You can use `"*"` to include all user attributes and [.codeph]`"+"` to include all operational attributes. **`--includeVirtualAttributes`** The exported entries include the values of virtual attributes in addition to real attributes. This tool can only be used over a secure connection and when authenticated as a user with the `permit-export-reversible-passwords` privilege. The server encrypts the output using a key generated from either a user-supplied passphrase or an encryption settings definition.

`extract-data-recovery-log-changes`	Extracts changes matching a given set of criteria from a PingDirectory 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). This tool is designed to be used in conjunction with the server's data recovery log files (in the `logs/data-recovery` directory). It can be used in conjunction with other audit log files, but for best results, the logger should be configured to operate in reversible form, to include the requester DN and IP address, and to include information about any intermediate client control that might have been provided in the request. This tool must not be used with a log file that the server can update while the tool is running, or that can have some content stored in an unwritten buffer (which is especially likely if the log is compressed or encrypted). To use this tool with the server online, you should only specify a log file that has already been rotated to ensure that no more writes will be made to that file. If necessary, use the **`rotate-log`** tool to force the current active file to be rotated. To use this tool to revert an inappropriate set of changes, run it with `--direction revert` and an additional set of arguments that identify which changes should be reverted (for example, based on the address of the client, the authorization DN of the requester, the time frame in which the changes were applied, etc.). To use this tool to replay changes that were previously applied (for example, after restoring an old backup or importing an old LDIF file), run it with `--direction replay` and an appropriate set of arguments to select the desired set of changes. Also, make sure to use **`dsreplication pre-external-initialization`** before performing the restore or import and applying the changes, and then use **`dsreplication post-external-initialization`** after the changes have been applied. See the PingDirectory Server Administration Guide for additional information. This tool will extract changes from the selected log file (and any previously rotated files, unless the `--doNotFollowRotationChain` argument is provided) and output them in LDIF change format. If the `--outputFile` argument is provided, then the changes will be written to that file; otherwise, they will be written to standard output. If changes are to be written to a file, then the output will be compressed if the input files were compressed (unless the `--doNotCompressOutput` argument was provided), and the output will be encrypted if the input files were encrypted (unless the `--doNotEncryptOutput` argument was provided). You might want to first run the tool without specifying an output file so that you can verify that the selected changes are correct. Once you are certain that the appropriate changes have been selected, you can use a tool like **`ldapmodify`** or **`parallel-update`** to apply those changes to the server.
`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 which 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. That is, it 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 a PingDirectory server backend. Connection parameters are not required when importing to a local PingDirectory server that is not running. However, connection parameters are required if the server is remote, or if it is running locally and it is inconvenient to have to stop it for the import. You can use the options to include or exclude specific attributes and branches of the tree, and to include or exclude entries matching a given filter. The input file can be compressed. This tool features both an offline mode of operation as well as the ability to schedule an operation to run within the PingDirectory 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.
`indent-ldap-filter`	Parse a provided LDAP filter string and display it a multi-line form that makes it easier to understand its hierarchy and embedded components. If possible, the tool 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.** **The `ldap-diff` tool outputs the difference between data stored in two LDAP servers into an LDIF file. This file could be used with the `ldapmodify` command to bring the source directory server in sync with the target directory server. The specific entries to compare can be controlled with the `--searchFilter` option. In addition, only a subset of attributes can be compared by listing those attributes as trailing arguments of the command. Specific attributes can also be excluded by prepending a `^` character to the attribute. On Windows operating systems, excluded attributes must be quoted. For example: `"^attrToExclude"`. When retrieving entries from a PingDirectory server, the `@objectClassName` notation can be used to compare only attributes that are defined for a given objectClass.** **This command can be used on servers actively being modified, without reporting false positives caused by replication delays, by checking differing entries multiple times. By default, it will re-check each differing entry twice, pausing two seconds between checks. These settings can be configured with the `--numPasses` and `--secondsBetweenPass` options. The output is formatted so that delete operations come first, modify operations come next, and add operations come last. This gives the best chance that the resulting output file can be used to bring the source server into sync with the target server without causing any conflicts. This takes into account attribute uniqueness constraints as well as the requirement that child entries be deleted before parents and parents be added before children.** **The directory 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 PingDirectory 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` **For servers from other vendors, consult their documentation for configuring the proper privileges.** **The `ldap-diff` tool tries to make efficient use of memory, but it must store the DNs of all entries in memory. For directories that contain tens 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 `ldap-diff` can be customized by editing the `ldap-diff.java-args` parameter in the `config/java.properties` file and running the `dsjavaproperties` command.**

`ldap-result-code`	Display and query LDAP result codes. This tool can be used to list all known defined LDAP result codes, retrieve the name of the result code with a given integer value, or search for all result codes with names containing a given substring. At most, one of the `--list`, `--int-value`, and `--search` arguments can be provided. If none of them is provided, then the `--list` option will be chosen by default.
`ldapcompare`	Perform compare operations in an LDAP directory server. Compare operations can be used to efficiently determine whether a specified entry has a given value. The exit code for this tool will indicate whether processing was successful or unsuccessful, and to provide a basic indication of the reason for unsuccessful attempts. By default, it will use an exit code of zero (which corresponds to the LDAP 'success' result) if all compare operations completed with a result code of either 'compare false' or 'compare true' (integer values 5 and 6, respectively), but if the `--useCompareResultCodeAsExitCode` argument is provided, only one compare assertion is performed, and it yields an exit code of 'compare false' or 'compare true', then the numeric value for that result code will be used as the exit code. If any error occurs during processing, then the exit code will be a nonzero value that reflects the first error result that was encountered. The attribute type and assertion value to use for the compare operations will typically be provided as the first unnamed trailing argument provided on the command line. It should be formatted with the name or OID of the target attribute type followed by a single colon and the string representation of the assertion value. Alternatively, the attribute name or OID can be followed by two colons and the base64-encoded representation of the assertion value, or it can be followed by a colon and a less-than symbol to indicate that the assertion value should be read from a file (in which case the exact bytes of the file, including line breaks, will be used as the assertion value). The DNs of the entries to compare can either be provided on the command line as additional unnamed trailing arguments after the provided attribute-value assertion, or they can be read from a file whose path is provided using the `--dnFile` argument. If the attribute-value assertion is provided on the command line as an unnamed trailing argument, then the same assertion will be performed for all operations. If multiple types of assertions should be performed, then you can use the `--assertionFile` argument to specify the path to a file containing both attribute-value assertions and entry DNs.
`ldapdelete`	Delete one or more entries from an LDAP directory server. You can provide the DNs of the entries to delete using named arguments, trailing arguments, a file, or 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. Unless the password change method is explicitly specified (using the `--passwordChangeMethod` argument), this tool will attempt to automatically determine which method is the most appropriate for the target server using information provided in the server's root DSE. If the server advertises support for the password modify extended operation, then that method will be used. If it appears to be an Active Directory server, then an Active Directory-specific password-change method will be selected, using a regular LDAP modify operation to update the `unicodePwd` attribute with a specially encoded value. Otherwise, a regular LDAP modify operation will be used to update the value of a specified password attribute. The new password to be set for the user can be specified in one of several ways. It can be directly provided on the command line, read from a specified file, interactively prompted from the user, or automatically generated by this tool. If the new password is not specified using any of those methods, and if the password is to be updated using the password modify extended operation, then the new password field of the request will be left blank so that the server generates a new password for the user and includes it in the response to the client. If no new password is specified and some other password change method is selected, then the tool will exit with an error. The current password for the user can also be specified. This is optional, although some servers might require a user to provide their current password when setting a new one. If a current password is provided (whether given as a command-line argument, read from a specified file, or interactively requested from the user), and if a regular LDAP modify operation is used to change the password, then the resulting modify request will include a deletion of the current value and an addition of the new value. If no current password is provided, then the modify request will replace any existing password(s) with the new value.
`ldapsearch`	Process one or more searches in an LDAP directory server. The criteria for the search request can be specified in several different ways, including providing all of the details directly using command-line arguments, providing all of the arguments except the filter using command-line arguments and specifying a file that holds the filters to use, or specifying a file that includes a set of LDAP URLs with the base DN, scope, filter, and attributes to return.

`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. This tool works best with small LDIF files because it reads the entire contents of the source and target LDIF files into memory so they can be quickly compared. If you encounter an out-of-memory error while running the tool, you might need to increase the amount of memory available to the JVM used to invoke it. The amount of memory available to the JVM can be customized by invoking the JVM with the `-Xms` and `-Xmx` arguments, which specify the initial and maximum amounts of memory that it can use, respectively. These arguments should be immediately followed, without any intervening space, by an integer and a unit to specify the amount of memory to be used. The unit can be either `m` to indicate that the size is in megabytes, or `g` to indicate that it is in gigabytes. For example, `-Xms512m` indicates that the JVM should be given an initial heap size of 512 megabytes, while `-Xmx2g` indicates that it should be given a maximum heap size of two gigabytes. When invoking the **`ldif-diff`** tool included in the installation of a Ping Identity server product, you can edit the `config/java.properties` file to specify the arguments to use when invoking the JVM. After modifying the file, run the **`dsjavaproperties`** tool to ensure that those changes will be used for subsequent tool invocations.
`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`** command cannot read the changes to apply from standard input. All of the change records will be read into memory before processing begins, so it is important to ensure that the tool is given enough memory to hold those change records. However, it will only operate on a single source entry at a time, so the size of the source LDIF file does not significantly impact the amount of memory that the tool requires. Note that the tool will attempt to correctly handle multiple changes affecting the same entry. However, because it only operates on one entry at a time, it cannot always behave in exactly the same way as if it were applying the changes over LDAP to a server populated with the source LDIF file. For example, it is not possible to reject an attempt to delete an entry that has subordinates, so any delete will be treated as a subtree delete. Further, not all types of modify DN change records are supported. In particular, modify DN change records are not permitted if they target any entry that has been targeted by a previous change record (for example, renaming an entry that was created by a previous add change record). Finally, it cannot perform other types of validation, like ensuring that all of the necessary superior entries exist when adding a new entry, or ensuring that a modify DN will not introduce a conflict with an existing entry.
`ldifsearch`	Search one or more LDIF files to identify entries matching a given set of criteria.

`leave-lockdown-mode`	Request that the PingDirectory server leave lockdown mode and resume normal operation. While in lockdown mode, the PingDirectory server rejects all requests from users that do not hold the lockdown-mode privilege. Note that the PingDirectory server can place itself in lockdown mode under certain conditions (for example, if it detects a security problem like a malformed access control rule that might have otherwise resulted in exposure of sensitive data).
`list-backends`	List the backends and base DNs configured in the PingDirectory server.
`load-ldap-schema-file`	Loads 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. See the server's `config/MakeLDIF` directory for example template files. 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 will be 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, PKCS #12, PKCS #11, or BCFKS key store.
`manage-extension`	Install or update PingDirectory server extension bundles. An extension bundle is a package of extension(s) that utilize the Server SDK to extend the functionality of the PingDirectory server. Extension bundles are installed from a zip archive or file system directory. The server will be restarted if running to activate the extension(s).
`manage-profile`	Generate, compare, install, and replace server profiles. Server profiles define a format for the configuration of a server, including `dsconfig`, initial DIT, setup arguments, server SDK extensions, and other files. These are combined into one concrete structure. This tool provides subcommands that can be used to generate a new profile from an existing server, to set up a new server, and to replace an existing server's profile with a different profile. A template server profile file structure can be found in the resource/ directory.
`manage-tasks`	Access information about pending, running, and completed tasks scheduled in the PingDirectory server.

`manage-topology`	Manage the topology registry. The topology registry is a branch of the configuration DIT (`cn=Topology,cn=config`). It stores all metadata about server instances, including their instance and listener certificates, secret keys, server groups and administrative user accounts. In addition, it also stores information about the replication topology (replication server ID and replication domain ID) when replication is enabled among servers in a Directory topology. Last but not least, it stores the license key required to install the server. Changes to the topology registry on one server are automatically mirrored to other servers in the topology. The **`dsconfig`** tool, configuration API, or the management console can be used to make changes to the topology registry. This tool allows some additional capability such as exporting the contents of the registry as a JSON file.
`migrate-ldap-schema`	Migrate schema information from an existing LDAP server into a PingDirectory server instance. This tool can be used to migrate schema information from an existing LDAP server into this PingDirectory server instance. The source server can be any standards-compliant LDAPv3 server. All attribute type and object class definitions, which are contained in the source LDAP server but not in the target PingDirectory server instance, will be either added to the target instance or written to a schema file.
`migrate-sun-ds-config`	Update an instance of the PingDirectory server to match the configuration of an existing Sun Java System Directory Server 5.x, 6.x, or 7.x. This tool can be used to compare the configuration of Sun Java System Directory Server 5.x, 6.x, or 7.x and PingDirectory server instances in order to identify any differences and update the PingDirectory server configuration to more closely match that of the Sun server instance.
`modrate`	Perform repeated modifications against an LDAP directory server.
`move-subtree`	Move all entries in a specified subtree from one server to another.
`oid-lookup`	Search the OID registry to retrieve information about items that match a given OID or name. The string to use to search the OID registry should be provided as an unnamed trailing argument. All items in the OID registry will be examined, and any items that contain the provided string in its OID, name, type, origin, or URL will be matched. If no search string is provided, the entire OID registry will be displayed.

`parallel-update`	Use multiple concurrent threads to apply a set of add, delete, modify, and modify DN operations read from an LDIF file. As with other tools like **`ldapmodify`**, changes in the LDIF file to be processed should be ordered such that if there are any dependencies between changes (for example, if one add change record creates a parent entry and another add change record creates a child of that parent), prerequisite changes come before the changes that depend on them. When this tool is preparing to process a change, it will determine whether the new change depends on any other changes that are currently in progress, and if so, will delay processing that change until its dependencies have been satisfied. If a change does not depend on any other changes that are currently being processed, then it can be processed in parallel with those changes. This tool will keep track of any changes that fail in a way that indicates they succeed if re-tried later (for example, an attempt to add an entry that fails because its parent does not exist, but its parent can be created later in the set of LDIF changes), and can optionally re-try those changes after processing is complete. Any changes that are not retried, as well as changes that still fail after the retry attempts, will be written to a rejects file with information about the reason for the failure so that an administrator can take any necessary further action upon them.
`populate-composed-attribute-values`	Populate entries in one or more backends with attribute values generated by one or more composed attribute plugins. This tool uses the configuration from a specified set of composed attribute plugin instances to identify which entries to update and what changes to apply. It can be used as an alternative to exporting the data to LDIF and re-importing to ensure that existing entries have an appropriate set of composed attribute values. This tool schedules an operation to run within the PingDirectory server's process. LDAP connection options must be supplied 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.
`profile-viewer`	View information in data files captured by the PingDirectory server profiler. Profiler data files are generated by the Profiler plugin. To create these data files, set the profile-action attribute of the Profiler configuration object to 'start' to begin collection. Set the `profile-action` attribute to `stop` to end collection and have the plugin write the file to `logs/profile`.{timestamp}.

`re-encode-entries`	Re-encode all or a specified portion of the entries in a local DB backend. This tool can be used to initiate a task that will cause a local DB backend to re-encode all or a specified subset of the entries that it contains. The contents of the entries will not be altered, but this provides a useful mechanism for applying significant changes to the way that entries are actually stored in the backend (for example, to apply encoding changes if a feature like data encryption or uncached attributes or entries is enabled). This tool schedules an operation to run within the PingDirectory server's process. LDAP connection options must be supplied 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.
`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. An index must be rebuilt if the database already contains data when the index is configured. The backend containing the provided base DN must be a local DB backend. The types of indexes that can be rebuilt include attribute indexes, VLV indexes, and JSON field indexes. **Note** PingDirectory does not support the rebuilding of system indexes, regardless of whether the rebuild is attempted online or offline. If you need to rebuild a system index, you must export the backend to LDIF and re-import it. This tool features both an offline mode of operation as well as the ability to schedule an operation to run within the PingDirectory 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.
`register-yubikey-otp-device`	Registers a YubiKey OTP device with the PingDirectory 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. This tool schedules an operation to run within the PingDirectory server's process. LDAP connection options must be supplied 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.
`remove-attribute-type-from-schema`	Safely remove an attribute type definition from the server schema. The tool will perform an appropriate set of validation before actually removing the attribute type from the schema. The below conditions must be satisfied before the attribute type can be removed. • The requester must have the update-schema privilege. • The attribute type must not be referenced by any other schema element. • The attribute type must not be defined in any schema file that is included with the PingDirectory server. Only custom attribute types can be removed from the schema. • The attribute type must not be referenced in the server configuration (for example, it must not be indexed by any backend). • The attribute type must not be present in any entry that exists in any backend. This tool schedules an operation to run within the PingDirectory server's process. LDAP connection options must be supplied 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.
`remove-backup`	Safely remove a backup from the specified PingDirectory server backend. This tool deletes the specified backup archive and updates the backup descriptor accordingly. As an alternative to removing a specific backup, you can automatically remove backups outside of specified count or age criteria. The `--retainFullBackupCount` argument can be used to indicate that the specified number of full backups should be retained, and any other full backups in the directory are eligible to be removed. The `--retainFullBackupAge` argument can be used to indicate that any full backups older than the specified age are eligible to be removed.
`remove-defunct-server`	Remove a server from this server's topology. This tool will remove the specified server from the topology. In general, the `uninstall` tool should be used to remove a server from the topology. The `remove-defunct-server` tool should only be used if a prior attempt to uninstall a server was unsuccessful or the system where the server was installed is no longer available, leaving the server permanently inaccessible from the topology. If the defunct server is online and is able to reach other servers in the topology, running remove-defunct-server from it will cleanly remove it from the topology. If it cannot reach the other servers, then remove-defunct-server must also be run from one of the online servers.

`remove-object-class-from-schema`	Safely remove an object class definition from the server schema. The tool will perform an appropriate set of validation before actually removing the object class from the schema. The below conditions must be satisfied before the object class can be removed. • The requester must have the update-schema privilege. • The object class must not be referenced by any other schema element. • The object class must not be defined in any schema file that is included with the PingDirectory server. Only custom object classes can be removed from the schema. • The object class must not be referenced in the server configuration. • The object class must not be present in any entry that exists in any backend. This tool schedules an operation to run within the PingDirectory server's process. LDAP connection options must be supplied 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.
`replace-certificate`	Replace the listener certificate for this PingDirectory server instance.
`restore`	Restore a backup of a PingDirectory server backend. Only one backend can be restored at a time by the restore command. The PingDirectory server should be stopped unless task connection options are supplied for a running server. You can list the backups contained in a particular backend backup directory. A backup taken on one system can be restored on another system. This tool features both an offline mode of operation as well as the ability to schedule an operation to run within the PingDirectory 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.
`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. If the file argument is provided one or more times to specify the target log file paths, then only those log files will be rotated. If the file argument is not given, then the server will trigger rotation for all supported log files. You must have the `config-read` and `config-write` privileges to run this tool, and you must have the necessary access control rights to create and monitor entries in the task backend. This tool schedules an operation to run within the PingDirectory server's process. LDAP connection options must be supplied 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.
`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, several 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 • 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 the **`search-logs`** 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 PingDirectory server process.
`set-delegated-admin-aci`	Request that the PingDirectory server assign appropriate ACI for configured delegated administrators of the Delegated Admin API.
`setup`	Perform the initial setup for a server instance. This tool features both interactive and non-interactive modes for accepting the product license terms and initially configuring a server instance.
`start-server`	Start the PingDirectory server.
`status`	Display basic server information. This tool prints information about the server, such as version, connection handlers, and data sources. Some information might not be available if the server is not running, or if authentication credentials are missing or do not have sufficient privileges, or if the invoking user does not have sufficient file system access rights.
`stop-server`	Stop or restart the server. This tool is used to stop or restart the local instance of the server (by omitting LDAP connection options), or a remote server (by interacting with it over LDAP). In addition, this tool is used to schedule the server for shutdown at a later time using the server's task interface.
`subtree-accessibility`	List or update the set of subtree accessibility restrictions defined in the PingDirectory server.
`sum-file-sizes`	Calculate the sum of the sizes for a set of files. This tool is used to find the sum of the sizes of one or more files. If any of the files specified is a directory then it will be recursively processed.
`summarize-access-log`	Examine one or more access log files to display several metrics about operations processed within the server.
`sync-pipe-view`	PingDataSync only: Display the detailed configuration of a sync pipe or pipes in PingDataSync and all commands necessary to replicate a specified sync pipe. You can use the `sync-pipe-view` tool online with the PingDataSync server connection credentials or you can use the tool offline. The sync pipe information can be output in a text, CSV, JSON, or tab-delimited format and can be output to a file. The `sync-pipe-view` tool features both an interactive mode and a non-interactive mode.
`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 the PingDirectory server. This tool removes the entire server or individual server components from the file system. If this server is a member of a replication topology, you must first remove references to this server in the other servers using the `dsreplication disable` command.
`update`	Update a deployed server so its version matches the version of this package.
`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 PingDirectory server. Note that output generated by this tool will be LDIF, but each entry in the output will have exactly one ACI, so entries which 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. The backend containing the provided base DN must be a local DB backend. The types of indexes that can be verified include system indexes, attribute indexes and VLV indexes. Any errors found during verification are written to the output. The verification process is exhaustive and can take a long time.
`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

The PingDirectory 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 the PingDirectory server in multiple environments.

Each property takes the form of a name-value pair that defines predetermined values for a tool's options.

The PingDirectory server supports the following types of properties:

- Default properties that apply to all command-line utilities
- Tool-specific properties

Evaluation of command-line options and file options

You can specify options for a command-line tool on the command line, in a properties file, or both.

Options you specify on a tool's command line take priority over options in a properties file.

Consider the following scenarios.

Command-line options	Server uses ...
No command-line options	The options in the default `<server-root>/config/tools.properties` file.
Command-line options other than the `--propertiesFilePath <my-properties-file>` option	The command-line options, which take priority if the options are also in the `<server-root>/config/tools.properties` file. The file options for options that are only in the default `<server-root>/config/tools.properties` file.
Only the `--propertiesFilePath <my-properties-file>` option	The options in `<my-properties-file>`.
The `--propertiesFilePath <my-properties-file>` option and other command-line options	The command-line options, which take priority if the options are also in `<my-properties-file>`. The file options for options that are only in `<my-properties-file>`.
The `--noPropertiesFile` option and other command-line options	Only the options you specify on the command line, ignoring the default properties file.

Example

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
```

The server checks command-line options and file options to determine the options to use, as explained below.

- All options presented with the tool on the command line take precedence over any options in a properties file.

  In the following example, the command runs with the options specified on the command line (`--port` and `--baseDN`). With the `port` value both on the command line and in the properties file, the command-line value takes priority. 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=*)"
  ```

- If you specify the properties file using the `--propertiesFilePath` option and no other command-line options, the server uses only the options in the specified properties file:

```
$ bin/ldapsearch --propertiesFilePath bin/tools.properties \
  "(objectclass=*)"
```

- If you do not specify any command-line options, the server attempts to locate the default properties file in the following location:

```
<server-root>/config/tools.properties
```

By moving your `tools.properties` file from `<server-root>/bin` to `<server-root>/config`, you do not have to specify the `--propertiesFilePath` option. That change shortens the previous command to the following command:

```
$ bin/ldapsearch "(objectclass=*)"
```

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 Lightweight Directory Access Protocol (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.

dsconfig batch files">

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

The PingDirectory 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.

Delegated Admin Application Guide

Delegated Admin is an add-on to PingDirectory that enables the delegation of user and group management.

Introduction to Delegated Admin

Delegated Admin is an add-on to PingDirectory that enables the delegation of user and group management.

Delegated Admin lets organizations assign responsibilities associated with the management of identities in the PingDirectory server to a subset of administrators.

These 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 and represent strong candidates for inclusion in a group of delegated administrators:

- 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

Delegated Admin Features

Delegated Admin lets delegated administrators complete tasks across groups, subtrees, and entire organizations.

Tasks include:

- 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 organizational unit (OU) records.
- Trigger a password reset process for a user that invokes the self-service password reset process defined by the business.
- Configure REST Resource Types to correlate to other resource types to create one-to-many relationships without schema changes. Edit or delete linked entries from the edit page of the primary entry.

Configuring Delegated Admin

This section describes the necessary configuration to support Delegated Admin after the application is installed successfully.

At a minimum, you must configure the following properties on the 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, see the documentation for each product.

The process of configuring support for Delegated Admin on a 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 a 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 the PingDirectory server as an OAuth token validator client.

Authentication configuration

The delegated administrator signs 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 the PingDirectory server, encapsulates information `claims` about the user's identity, and issues an access token to Delegated Admin, which presents the token to the PingDirectory server in the HTTP Authorization request header.

Interaction with the PingDirectory server

The PingDirectory server is configured to accept access tokens by using access token validators. The values that the PingFederate server sets for the access token `sub` claim must be mappable to a distinguished name (DN) in the 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 the PingDirectory server

After validation, the PingDirectory server checks the Delegated Admin configuration for authorization of the delegated administrator. Users or groups of users are authorized as delegated administrators in the PingDirectory server admin 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 Configuring Delegated Admin as a new client (create OAuth client for Delegated Admin).
- The PingDirectory server itself, which calls PingFederate to validate the OIDC token that Delegated Admin passes to it. For more information, see Configuring the PingDirectory server as the token validator (create OAuth client for PingDirectory).

Configuring delegated administrator rights on the PingDirectory server

To delegate users or groups as administrators, use the PingDirectory admin console (Delegated Admin rights and resource rights) or the `dsconfig create-delegated-admin-rights` and `create-delegated-admin-resource-rights` commands.

About this task

To use Delegated Admin, an administrator must possess rights that are designated through the PingDirectory server configuration in addition to valid credentials and an access token that the PingDirectory server can validate.

Admin Permissions

`create`

The administrator can create new resources of this type.

`read`

The administrator can read resources of this type.

> **Note**
>
> The `create`, `delete`, `update`, `update-profile`, `reset-password`, and `manage-group-membership`, `update` permissions require the `read` permission.

`update`

The administrator can edit resources of this type.

`delete`

The administrator can delete resources of this type.

`update-profile`

The administrator can update user profiles but isn't allowed password- change-related privileges.

For group and generic type resources, the `update-profile` permission gives the same rights as the `update` permission.

`reset-password`

The administrator can reset passwords without the ability to change other user attributes.

`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.

`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 distinguished name (DN) without displaying the actual values of the DNs.

`download`

The administrator can download reports for resources of this type. With this permission, the Download Report button shows 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 shows 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 can't be changed without invalidating the configuration. The attribute label has a lock icon and a message indicating that the value can only be changed by a server administrator.

The example commands that follow illustrate the configuration options for delegated administration and are performed on the PingDirectory server.

> **Note**
>
> Administrators who manage only specific subtrees can't create users in an organization that does not reside under, or at the same level as, one of the subtrees.

Steps

- **Restrict an administrator to manage users in specified subtrees.**

  Example:

```
$ 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
```

- **Restrict an administrator to managing the member users of one or more specified groups.**

  Example:

  In the following example, 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
```

- Assign the delegated admin 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, delegate an entire group of users as administrators.

  For more information about the PingDirectory server administrators and configuring dynamic and static groups, see the PingDirectory Server Administration Guide.

  Example:

  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`.

```
$ 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
```

Parameterized Delegated Administrator Rights

Delegated Admin rights can be parameterized so that a single definition provides a pattern for new administrators.

This allows a privileged administrator for a hosting company to 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 the PingDirectory server configuration changes to create a new administrator.

In the following example, it's assumed that there are three REST resource types configured:

- `orgs`

- `groups`
- `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.
- Add 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 sign on to the app and manage users for their own organization.

Configuring user self-service

The PingFederate server provides end users with the ability to self-service their own profiles.

About this task

To enable users created by delegated administrators to manage their own profiles through the PingFederate local identity profile-management feature, you need to perform additional configuration steps in both PingDirectory and PingFederate.

> **Note**
>
> Import the PingFederate LDAP Data Interchange Format (LDIF) first in PingDirectoryProxy and then in PingDirectory. Constructed attributes need to be created only in PingDirectoryProxy. Creating and rebuilding indexes (part of the self-service configuration) is done on PingDirectory.

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 in the PingFederate documentation.

Setting up PingDirectory for customer identities in the PingFederate documentation includes some of the following required steps on the 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.

   Example:

   This policy requires that you enable Allow Password Changes as mentioned above.

   ```
   dsconfig set-password-policy-prop --policy-name "Default Password Policy" \
   --set force-change-on-add:true --set force-change-on-reset:true
   ```

   Result:

   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 Lightweight Directory Access Protocol (LDAP) schema provided by PingFederate into PingDirectory.

   1. On the 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`.
   2. Use the `scopy` command to securely copy the LDIF file to your local machine.

4. Update the LDAP schema.

   1. Sign on to the PingDirectory admin console.
   2. Go to LDAP Schema → Schema Utilities.
   3. Click Import Schema Element.
   4. Copy the schema changes from the file `<pf_install>/pingfederate/server/default/conf/local-identity/ldif-scripts/local-identity-pingdirectory.ldif`.
   5. Paste the schema changes into the text area.
   6. 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.

Example:

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 is 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 distinguished name (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) 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 the PingDirectory server

Use the Delegated Admin installation file to configure attributes and attribute search.

About this task

The file that installs Delegated Admin 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.

   Example:

   ```
   $ 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 Lightweight Directory Access Protocol (LDAP) attribute type to expose and by providing a display name for it.

   Example:

   ```
   $ bin/dsconfig create-delegated-admin-attribute \
     --type-name users \
     --attribute-type customAttr \
     --set "display-name:My custom attribute"
   ```

3. Configure attributes with distinguished name (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 doesn't 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.

   Example:

   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
   ```

   Example:

   Additionally, 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 the PingDirectory server.

4. Use the following command to set the search filter, where `%%` represents the search text entered in the web application.

   Example:

   ```
   $ 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 the PingDirectory server. To satisfy the query, define the appropriate attribute indexes for the PingDirectory server. For more information, see the 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.

   Example:

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.

   Example:

   ```
   $ 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**
   >
   > This setting does not apply to attributes that are handled by custom UI form fields.

   Example:

   ```
   $ 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 created or 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**
>
> An attribute's capability of being changed after its creation is called its mutability.

As with standard attributes, constructed attributes are stored as LDAP attributes in a database like the PingDirectory server.

Setting an attribute to read-only

About this task

Beginning with Delegated Admin 3.5.0 and PingDirectory 7.3.0.1, you can set user access to standard and constructed attributes to `read-only` and `read/write`. You should 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.

Steps

- Use the `dsconfig` tool to set a standard or constructed attribute as `read-only`.

  Example:

  ```
  dsconfig set-delegated-admin-attribute \
    --type-name users  \
    --attribute-type modifyTimestamp  \
    --set mutability:read-only
  ```

  Example:

  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

You can configure delegated administrators to manage users and groups in the PingDirectory server.

Configuring delegated administrators involves the following:

- Create new entries.
- Read, view, and search existing entries.
- Edit and update existing entries.
- Edit referenced items.

  For example, if a delegated administrator is viewing an item that references a user, the delegated administrator can edit that user's information inline.

The following sections describe users and groups in more detail.

Enabling 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 the 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, see 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. 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 the 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 the 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 the PingDirectory server. 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 Lightweight Directory Access Protocol (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 can't sign on to Configuring user self-service.

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 displays.

Steps

1. For each PingDirectory instance that contains users, enable the `Password Policy State JSON` virtual attribute for the users object class.

   > **Note**
   >
   > You don't 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`, such as `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.` (This entry also displays when the user hasn't logged in.)

PASSWORD EXPIRATION

`Password expiration date has not been enabled.`

You can configure these items in the admin console or by using the `dsconfig` tool interactively or non-interactively. The following steps use the non-interactive approach:

2. Select the password policy for which you want to enable the last login time and password expiration date.

   Learn more about Managing password policies.

   ```
   dsconfig list-password-policies
   ```

3. (Optional) Include the last login time.

   To include the last login time, choose 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

   Learn more about last login time options.

4. (Optional) Include the password expiration date.

   To include this information, set the property `max-password-age`.

5. Set the desired password policy properties.

   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
   ```

Setting up initiate password reset for REST resource types

To initiate a password reset, a given REST resource type must have the `ds-pwp-modifiable-state-json` delegated admin attribute.

About this task

> **Note**
>
> The `ds-pwp-modifiable-state-json` delegated admin attribute is not visible on the **View/Edit** and **Reporting** pages. It's for internal use only, similar to the `ds-pwp-account-disabled` attribute.

To enable initiate password reset functionality for a specified REST resource type:

Steps

- Run `dsconfig` with the `create-delegated-admin-attribute` option.

  Example:

  The following example grants `"Tenant Users"` the initiate password reset functionality through the `ds-pwp-modifiable-state-json` delegated admin attribute:

  ```
  dsconfig create-delegated-admin-attribute \
      --type-name "Tenant Users"  \
      --attribute-type ds-pwp-modifiable-state-json  \
      --set "display-name:Modifiable Password Policy State"
  ```

Managing groups

The administrative scope for users determines which users are visible to the group administrator.

You can use `dsconfig` to delegate a user as a group administrator. An administrator can be configured to edit users and manage group memberships. When configuring an administrator, consider the following:

- 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 groups of which they are a member. For example, an Employees group might include a Developers group as a nested member. In this 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.

Example

In the following example, all users in the subtree `ou=org1,dc=example,dc=com` are visible:

```
$ 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
```

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=%%)'
```

Rename the Members and Nonmembers columns

An administrator can assign custom names for Members and Nonmembers columns in the Delegated Admin application for the following rest resource types:

- Groups
- Users
- Generic rest resource types

To set these column titles, run `dsconfig set-rest-resource-type-prop`:

```
dsconfig set-rest-resource-type-prop \
    --type-name users \
    --set "members-column-name:<custom member label>" \
    --set "nonmembers-column-name:<custom non member label>"
```

Viewing groups

Use the group `reference` right to grant delegated admins the right to see what groups a user is a member of without granting group resource management rights.

About this task

To grant a delegated admin the group `reference` right:

Steps

- Run `dsconfig` with the `create-delegated-admin-resource-rights` option.

```
dsconfig create-delegated-admin-resource-rights \
    --rights-name DArights  \
    --rest-resource-type groups  \
    --set enabled:true  \
    --set admin-permission:reference  \
    --set admin-scope:all-resources-in-base
```

Result

The admin can see the user's group memberships on the entry preview's Group Membership tab.

Creating a group

You can add users as members to groups that delegated administrators create and manage. You can also add subgroups 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 distinguished name (DN) for a resource type, enter 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 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

These attributes are presented to the delegated administrators.

To configure a group REST resource type, go to Configuration > REST Resource Types page in the PingDirectory admin console.

When creating or editing a REST resource type, the Search Base DN field in the General Configuration section determines the data structure that is searched in Delegated Admin, and the Display Name field in the Delegated Admin section specifies the label of the REST resource in the Delegated Admin GUI.

PingDirectory Attributes

PingDirectory admin console		Delegated Admin GUI
UI field	Window and section	UI field onNew Group page
`Display Name`	Configuration > REST Resource Types. Create or edit a REST resource type, and then go to the Delegated Admin section.	`Select a Type` label
`REST Resource Type`	Configuration > Delegated Admin Rights. Create or edit Delegated Admin rights, and then click New Delegated Admin Resource Rights.	`Select a Type` option
`Parent Resource Type`	Configuration > REST Resource Types. Create or edit a REST resource type, and then go to the Resource Creation section.	Display name for parent resource type
`Display Name`	Configuration > REST Resource Types. Create or edit a REST resource type, and then go to the Delegated Admin Attributes section. Click New Delegated Admin Attribute.	Additional fields such as `CN`, `Description`, `Business Category`, and `Organization`

Adding a user to a group

You can add users to groups from the Manage Users page and from the Manage Groups page in the Delegated Admin GUI.

Adding a new user to a configured group

About this task

When the delegated admin rights for a user REST resource type have the `resources-in-specific-groups` admin scope, a user is added to one of the configured groups when you create the user:

- For admins with rights to only one group, the new user is automatically added to that group. No field for `Select Group` displays.
- For admins with 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, in a dynamic group of members with `uid=user.111*`, the `uid` starts with `user.111`.

To create a new user in that group, an administrator must:

Steps

1. In the `Select Group` list, select the group name.
2. Enter the value for `uid` that starts with `user.111`.

Adding a user from the Manage Users window

You can add users to groups using the Manage Users window in the Delegated Admin GUI.

Steps

1. In the Delegated Admin GUI, go to the Manage Users window.
2. Use to search field to search for the user to add to a group.
3. Click the Expand icon on the appropriate user profile.
4. To edit the user profile, click the Pencil icon.
5. Click the Group Membership tab.
6. Use the search field to search for the appropriate group.
7. To add the user to the respective group, in the Nonmember Groups list, click +.

   Result:

   The group moves to the Member Groups list.

Adding a user from the Manage Groups window

You can add users to groups using the Manage Groups window in the Delegated Admin GUI.

Steps

1. In the Delegated Admin GUI, go to the Manage Groups window.
2. To filter by group resource type, make a selection from the drop-down list for the group resource type you want to search within.
3. (Optional) To narrow your results, use the search field to search for the appropriate group.
4. Click the Expand icon on your chosen group.
5. To edit the group profile, click the Pencil icon.
6. Under the group name, click Group Membership.
7. (Optional) To filter entries by resource type, make a selection from the Select a Type drop-down list.

   > **Note**
   >
   > If there is a parent resource type, a second drop-down list appears. Make an additional selection, or leave it to **All** to see all entries in that resource type.

8. Use the search field to search for the appropriate entry.
9. To add an entry to the group, from the Nonmembers list, click +.

   Result:

   The user moves to the Members list.

Unlocking user accounts

To unlock a user account without requiring a password reset, use the Delegated Admin application.

About this task

> **Note**
>
> To require a password reset when unlocking a user, use the **Reset Password** or **Set Password** options in the Delegated Admin application.

To use Delegated Admin to unlock a user account without requiring a password reset:

Steps

1. In the Delegated Admin application, go to Manage Users.
2. In the Search box, enter the user ID of the user you need to unlock and press Enter.
3. (Optional) Click the Expand icon to expand the user entry.
4. Click the Locked toggle.

   Result:

   The toggle turns green and changes to Enabled.

Result

The user can access their account without resetting their password.

Enabling the Delegated Admin user REST resource type photo upload feature

In Delegated Admin 4.10.0 and later, you can add a photo to a user REST resource type profile in the Delegated Admin application.

To use the Delegated Admin photo upload feature, you must first enable the feature. You can use the PingDirectory admin console or `dsconfig` to perform this task.

After you have enabled the feature, it's ready to use when creating a new user or when editing an existing user. To use the enabled photo upload feature, see Uploading a photo to a user REST resource type profile in Delegated Admin.

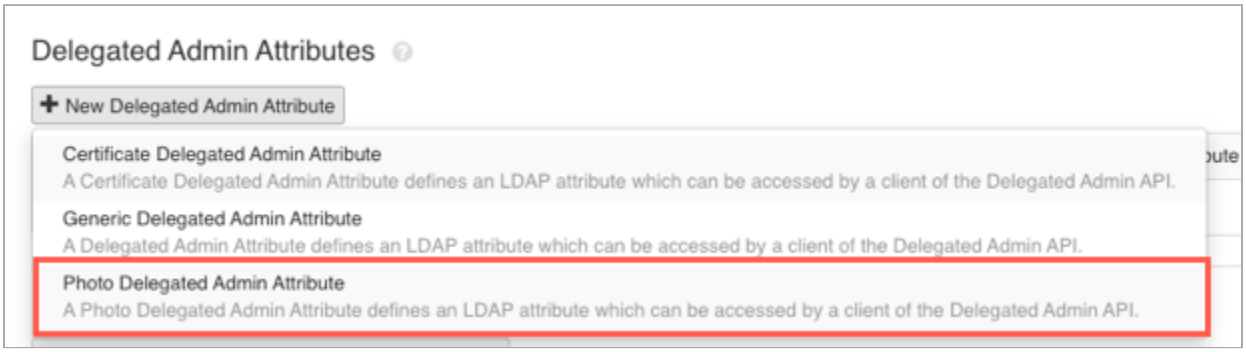
Enabling the user profile photo upload feature using the admin console

About this task

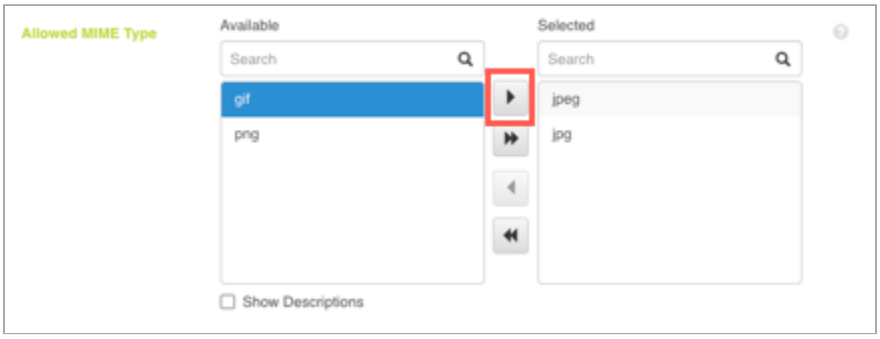
To enable the Delegated Admin user REST resource type photo upload feature using the PingDirectory admin console:

Steps

1. In the Web Services and Applications list, select REST Resource Types.
2. In the REST Resource Types list, select your user REST resource type.
3. Go to Delegated Admin Attributes and click New Delegated Admin Attribute.
4. In the New Delegated Admin Attribute list, select Photo Delegated Admin Attribute.



5. In the Attribute Type field, enter the attribute type, such as `jpegPhoto`.
6. Enter a Display Name for the attribute.
7. Keep the default value for Mutability.
8. (Optional) To allow multiple files to be uploaded for one user, enable Multi Valued.
9. Enter a Display Order Index or keep the default, `0`.
10. In the Allowed MIME Type list, select your desired allowed MIME file types from the Available column and use the arrows to add your selections to the Selected column.



11. Click Save.

Enabling the user profile photo upload feature using `dsconfig`

About this task

> **Tip**
>
> To allow multiple files to be uploaded for one user, set `multi-valued` to `true`, as in the second example.

To enable the Delegated Admin user REST resource type photo upload feature using `dsconfig`:

Steps

- Run `dsconfig create-delegated-admin-attribute`.

  Example:

  The following example creates a `jpegPhoto` Delegated Admin attribute with `.png` and `.jpg` set as accepted file types:

  ```
  bin/dsconfig create-delegated-admin-attribute \
      --type-name users  \
      --attribute-type jpegPhoto  \
      --type photo \
      --set display-name:Photo  \
      --set display-order-index:3  \
      --set allowed-mime-type:png  \
      --set allowed-mime-type:jpg  -n
  ```

  Example:

  The following example creates a multivalued `jpegPhoto` Delegated Admin attribute with `.png` and `.jpg` set as accepted file types:

```
bin/dsconfig create-delegated-admin-attribute \
    --type-name users  \
    --attribute-type jpegPhoto  \
    --type photo \
    --set display-name:Photo  \
    --set multi-valued:true  \
    --set display-order-index:3  \
    --set allowed-mime-type:png  \
    --set allowed-mime-type:jpg  -n
```

Uploading a photo to a user REST resource type profile in Delegated Admin

After you have enabled the photo upload feature, you can use it to add a photo to a new or existing user REST resource type profile in Delegated Admin.

> **Important**
>
> You must enable the photo upload feature before you can use it in the Delegated Admin application. For more information and to enable the feature, see Enabling the Delegated Admin user REST resource type photo upload feature.

Uploading a photo to a new user profile in Delegated Admin

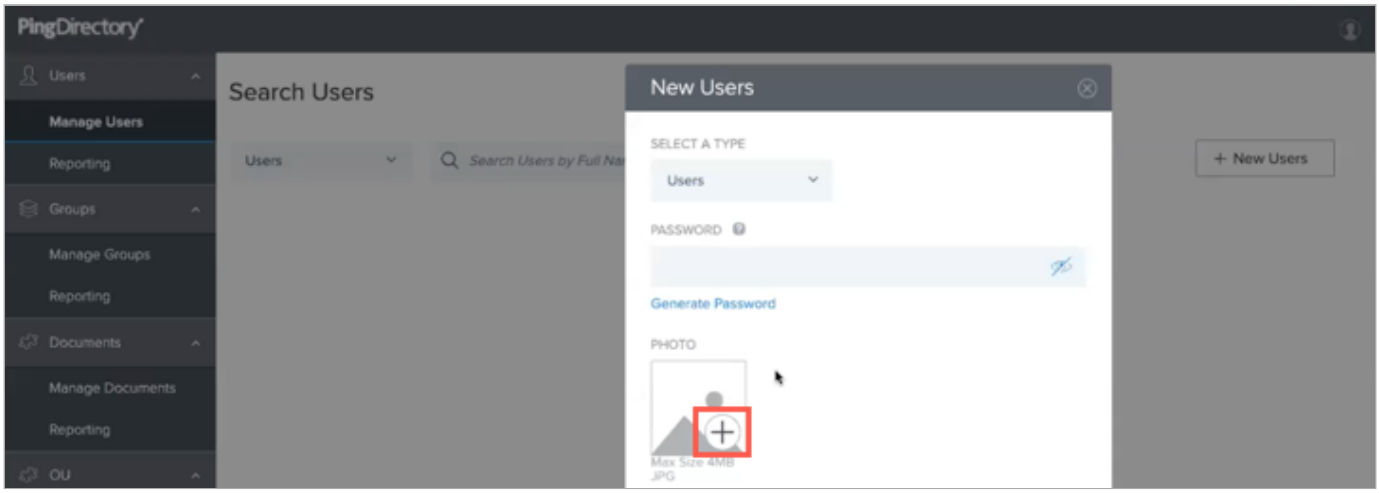
About this task

To add a photo to a new user profile in the Delegated Admin application:

Steps

1. Click Users → Manage Users.
2. Click New Users.

3. To add a photo, click the + icon and select a photo.



> **Note**
>
> Because of a server limitation, only a thumbnail of the photo and not the file name is displayed after the upload.
> The maximum photo size is 4MB.

4. Complete all other required fields for the new user.
5. Click Save.

Uploading a photo to an existing user profile in Delegated Admin

About this task

To add a photo to an existing user profile in the Delegated Admin application:

Steps

1. Click Users → Manage Users.
2. To locate the user, enter the user information in the search field.
3. Click the Expand icon on the user profile.
4. To edit the profile, click the Pencil icon.
5. (Optional) If you already have a photo and want to replace it, click Remove to remove the original photo.
6. To add a photo, click the + icon and select a photo.

> **Note**
>
> Because of a server limitation, only a thumbnail of the photo and not the file name is displayed after the upload.
> The maximum photo size is 4MB.

7. Click Save.

Enabling the Delegated Admin user REST resource type certificate upload feature

In Delegated Admin 4.10.0 and later, you can add a certificate to a user REST resource type profile in the Delegated Admin application.

To use the Delegated Admin certificate upload feature, you must first enable the feature. You can use the PingDirectory admin console or `dsconfig` to perform this task.

After you've enabled the feature, it's ready to use when creating a new user or when editing an existing user. To use the enabled certificate upload feature, see Uploading a certificate to a user REST resource type profile in Delegated Admin.

Enabling the user profile certificate upload feature using the admin console

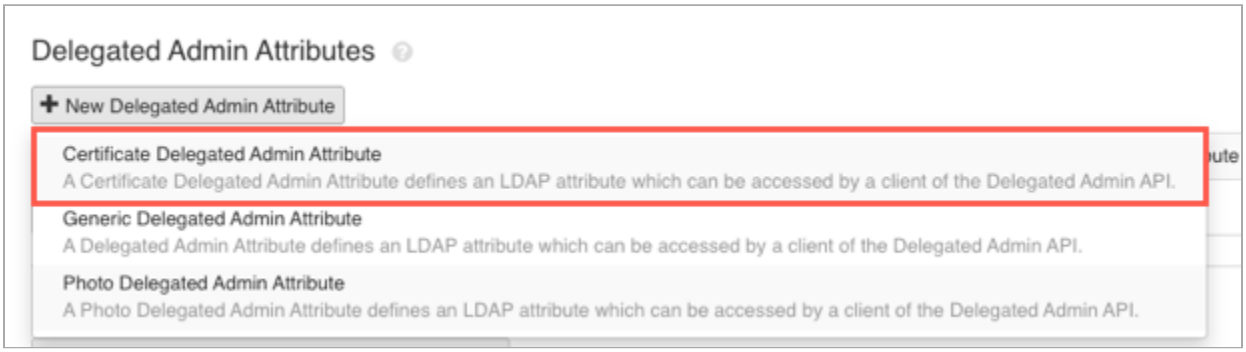
In Delegated Admin 4.10.0 and later, the user profile certificate upload feature can be enabled using the PingDirectory admin console.

About this task

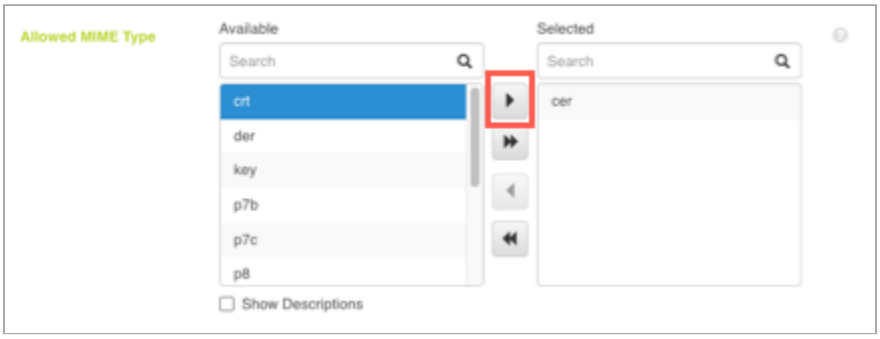
To enable the Delegated Admin user REST resource type certificate upload feature using the PingDirectory admin console:

Steps

1. In the Web Services and Applications list, select REST Resource Types.
2. In the REST Resource Types list, select your user REST resource type.
3. Go to Delegated Admin Attributes and click New Delegated Admin Attribute.
4. In the New Delegated Admin Attribute list, select Certificate Delegated Admin Attribute.



5. In the Attribute Type field, enter the attribute type, such as `userCertificate`.
6. Enter a Display Name for the attribute.
7. Keep the default values for Mutability.
8. (Optional) To allow multiple files to be uploaded for one user, enable Multi Valued.
9. Enter a Display Order Index or keep the default, `0`.
10. In the Allowed MIME Type list, select your desired allowed MIME file types in the Available column, and use the arrows to add your selections to the Selected column.



11. Click Save.

Enabling the user profile certificate upload feature using `dsconfig`

In Delegated Admin 4.10.0 and later, the user profile certificate upload feature can also be enabled using `dsconfig`.

About this task

> **Tip**
>
> To allow multiple files to be uploaded for one user, set `multi-valued` to `true`, as in the second example.

To enable the Delegated Admin user REST resource type profile certificate upload feature using `dsconfig`:

Steps

- Run `dsconfig create-delegated-admin-attribute`.

  Example:

  The following example creates a `userCertificate` attribute with `.cer` and `.crt` set as accepted file types:

  ```
  bin/dsconfig create-delegated-admin-attribute \
      --type-name users  \
      --attribute-type userCertificate  \
      --type certificate  \
      --set "display-name:user certificate"  \
      --set display-order-index:3  \
      --set allowed-mime-type:cer  \
      --set allowed-mime-type:crt -n
  ```

  Example:

  The following example creates a multivalued `userCertificate` attribute with `.cer` and `.crt` set as accepted file types:

```
bin/dsconfig create-delegated-admin-attribute \
    --type-name users  \
    --attribute-type userCertificate  \
    --type certificate  \
    --set "display-name:user certificate"  \
    --set multi-valued:true  \
    --set display-order-index:3  \
    --set allowed-mime-type:cer  \
    --set allowed-mime-type:crt -n
```

Uploading a certificate to a user REST resource type profile in Delegated Admin

After you have enabled the certificate upload feature, you can use it to add a certificate to a new or existing user REST Resource type profile in Delegated Admin.

> **Important**
>
> You must enable the certificate upload feature before you can use it in the Delegated Admin application. For more information and to enable the feature, see Enabling the Delegated Admin user REST resource type certificate upload feature.

Uploading a certificate to a new user profile in Delegated Admin

You can upload a certificate to a new user profile in the Delegated Admin application.

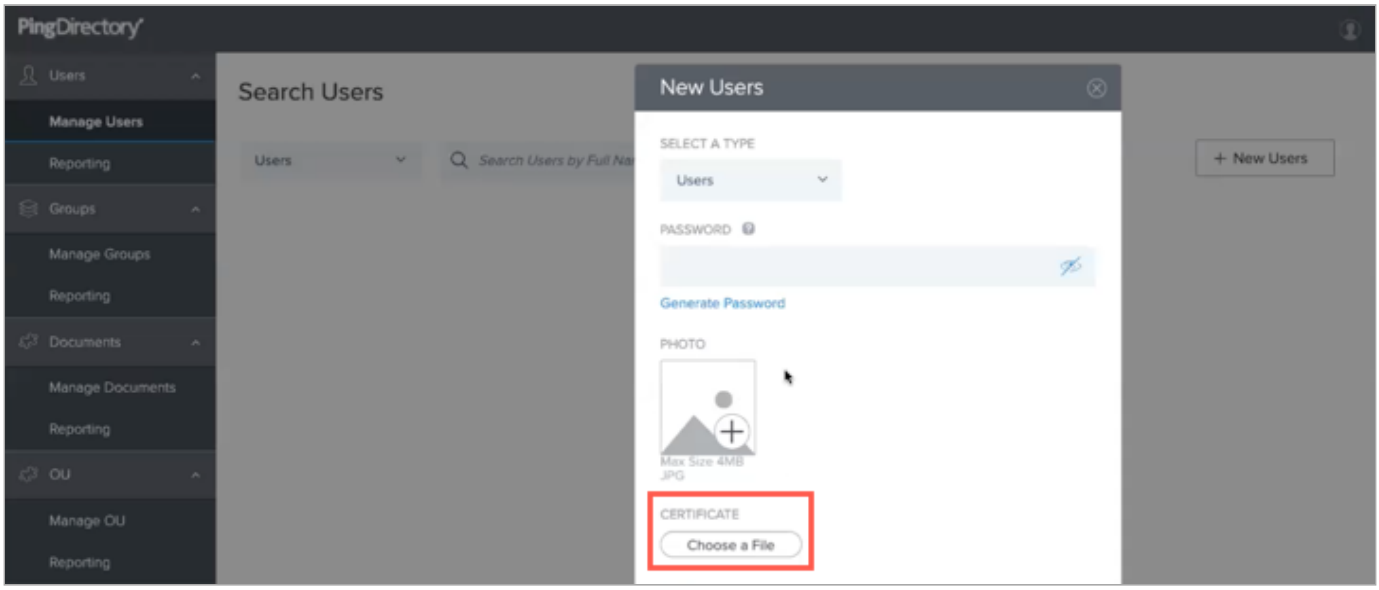
About this task

To add a certificate to a new user profile in the Delegated Admin application:

Steps

1. Click Users → Manage Users.
2. Click New Users.

3. To add a certificate, click Choose a File.



4. Complete all other required fields for the new user.
5. Click Save.

Uploading a certificate to an existing user profile in Delegated Admin

You can upload a certificate to an existing user profile in the Delegated Admin application.

About this task

To add a certificate to an existing user profile in the Delegated Admin application:

Steps

1. Click Users → Manage Users.
2. To locate the user, enter the user information in the search field.
3. Click the Expand icon on the user profile.
4. To edit the profile, click the Pencil icon.
5. To add a certificate, click Choose a File and select a certificate file.
6. Click Save.

Generic resource types

Use generic resource types to manage Lightweight Directory Access Protocol (LDAP) entries that are neither users nor groups.

You can perform the following actions with generic resource types:

- Create new entries.
- Read, view, and search existing entries.

- Edit and update existing entries.
- Edit referenced items.

For example, a generic resource type can represent a device or used for organizational unit branch entries that are parents of other resources.

Defining a generic resource type

You can define generic resources for any structural LDAP object class and they can function as members of a group.

Steps

1. Define a generic resource type.

   The following example enables the management of device entries:

   Example:

```
$ 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
```

2. Create Delegated Admin resource rights for the generic resource type.

   Generic resource administrators must have read access to the user resource. Learn more about Configuring delegated administrator rights on the PingDirectory server.

Working with correlated REST resources

You can link resource types based on a common attribute value. When two or more resources are linked this way, they are known as correlated resources.

Correlated REST resources

When you retrieve a resource configured with correlated resources, any correlated resource with matching attribute values is included with the resource.

You can view, update, and delete a resource's correlated resources. You can also link them to and unlink them from a resource.

Configuration

The following example shows how to create a correlated resource by linking a `documents` resource type to a `users` resource type. When a user resource is retrieved, any document resources with the user's `entryUUID` in their `documentPublisher` attribute are included.

```
dsconfig create-delegated-admin-correlated-rest-resource \
    --type-name users  \
    --resource-name Documents  \
    --set display-name:Documents  \
    --set correlated-rest-resource:Documents  \
    --set primary-rest-resource-correlation-attribute:entryUUID  \
    --set secondary-rest-resource-correlation-attribute:documentPublisher
```

Linking and unlinking

You can link resources to correlated resources by taking an attribute value from either the primary resource or the correlated resource and assigning it to the other. To control the direction that the linking value is assigned, use the `use-secondary-value-for-linking` property. If `use-secondary-value-for-linking` is `false` or isn't set, the primary correlation attribute's value is assigned to the secondary correlation attribute to create the link. If `use-secondary-value-for-linking` is `true`, the secondary correlation attribute's value is assigned to the primary correlation attribute.

The following is an example of a `users` resource type linked to a `documents` resource type by assigning the document's `entryUUID` value to the user's `description` attribute. The secondary correlation attribute's value is assigned to the primary correlation attribute, so `use-secondary-value-for-linking` is `true`.

```
dsconfig create-delegated-admin-correlated-rest-resource \
    --type-name users  \
    --resource-name ownedDocuments  \
    --set display-name:Owned Documents  \
    --set correlated-rest-resource:Documents  \
    --set primary-rest-resource-correlation-attribute:description  \
    --set secondary-rest-resource-correlation-attribute:entryUUID  \
    --set use-secondary-value-for-linking:true
```

If you inspected a user's `description` attribute with this setup, you see a list of all of the documents owned by the user.

The next example links the attributes in the other direction. Here, the user's `entryUUID` value is assigned to the `documentAuthor` attribute on the document.

```
dsconfig create-delegated-admin-correlated-rest-resource \
    --type-name users  \
    --resource-name authoredDocuments  \
    --set display-name:Authored Documents  \
    --set correlated-rest-resource:Documents  \
    --set primary-rest-resource-correlation-attribute:entryUUID  \
    --set secondary-rest-resource-correlation-attribute:documentAuthor  \
    --set use-secondary-value-for-linking:false
```

If you inspect a document's `documentAuthor` attribute with this setup, you see a list of all of the users who authored the document.

You can also unlink correlated resources from the primary resource. Resources are unlinked by removing the common attribute value using the logic based on the `use-secondary-value-for-linking` property described above.

Unlinking correlated resources differs from deleting correlated resources. When correlated resources are unlinked, the common attribute value is removed, but the correlated resource still exists in the directory. When a correlated resource is deleted, it is removed from the directory.

The attribute that the linking value is assigned to must be a writable attribute because the value must be updated to link and unlink correlated resources. The attributes should also have the same syntax so that the value of one can be assigned to the other successfully.

> **Note**
>
> When `use-secondary-value-for-linking` is enabled and a linked object is deleted, the linking value is not removed from the primary correlation attribute. This means that if a secondary object is recreated with the same value in the secondary correlation attribute, the secondary object is automatically linked to the primary object.

Permissions

The following table describes the permissions required to perform the specified actions.

Action	Permissions required
View	The `read` permission on the secondary resource type.
Update	The `update` or `update-profile` permission on the secondary resource type.
Delete	The `delete` permission on the secondary resource type.

Action	Permissions required
Link or unlink	The `update` or `update-profile` permission. **Important** The required permission is dependent on the `use-secondary-value-for-linking` setting. If `use-secondary-value-for-linking` is set to `false`, the permission is required on the correlated resource type. If `use-secondary-value-for-linking` is set to `true`, the permission is required on the primary resource type.

Setting up a DN reference attribute

A REST resource type can reference or edit other REST resource types based on a distinguished name (DN) without the full DN value. The following task, performed by the system administrator, grants the Delegated Admin administrator this ability.

About this task

In this task, the Delegated Admin attribute used for reference must be in DN syntax, such as `manager`, `entryDN`, or `seeAlso`.

Steps

1. To use a resource type for DN reference:

   Choose from:

   - Use an existing REST resource type.
   - Create a new REST resource type.

2. To add a `reference` Delegated Admin attribute, run `dsconfig` with the `create-delegated-admin-attribute` option using the following syntax.

   Example:

   ```
   dsconfig create-delegated-admin-attribute \
       --type-name users \
       --attribute-type <attribute of DN syntax> \
       --set "display-name:<display name>" \
       --set display-order-index: <index number> \
       --set reference-resource-type:<rest-resource-type>
   ```

   Example:

   The following example uses the values from Creating and configuring a new REST resource type:

```
dsconfig create-delegated-admin-attribute \
    --type-name users \
    --attribute-type manager \
    --set "display-name:Select Manager" \
    --set display-order-index:4 \
    --set reference-resource-type:Managers
```

Creating and configuring a new REST resource type

You can create and configure a new REST resource type in order to set up a reference delegated admin attribute.

About this task

To set up a reference delegated admin attribute, you must have a REST resource type defined and configured.

This example task creates a new REST resource type called `Managers`.

Steps

1. To create the new REST resource type, run `dsconfig` with the `create-rest-resource-type` option.

   Example:

   In the following example, the new resource type of `Managers` is created:

   ```
   dsconfig create-rest-resource-type \
       --type-name Managers \
       --type user \
       --set 'description:Rest type for users who are managers\n\It is used as reference type for the
   field Manager.' \
       --set enabled:true \
       --set resource-endpoint:managers \
       --set structural-ldap-objectclass:inetOrgPerson \
       --set search-base-dn:dc=example,dc=com \
       --set "include-filter:(employeeType=manager)" \
       --set parent-dn:ou=people,dc=example,dc=com \
       --set create-rdn-attribute-type:uid \
       --set display-name:Managers \
       --set 'search-filter-pattern:(|(cn=*%%*)(mail=%%*)(uid=%%*)(sn=*%%*))' \
       --set primary-display-attribute-type:cn
   ```

2. To add Delegated Admin attributes for the resource type, run `dsconfig` with the `create-delegated-admin-attribute` option.

   Example:

   The following example adds Delegated Admin attributes for the `Managers` resource type:

```
dsconfig create-delegated-admin-attribute \
    --type-name Managers \
    --attribute-type cn \
    --set "display-name:Full Name"

dsconfig create-delegated-admin-attribute \
    --type-name Managers \
    --attribute-type uid \
    --set "display-name:Manager ID"

dsconfig create-delegated-admin-attribute \
    --type-name Managers \
    --attribute-type mail \
    --set display-name:Email

dsconfig create-delegated-admin-attribute \
    --type-name Managers \
    --attribute-type employeeType \
    --set "display-name:Employee Type (must be manager)"

dsconfig create-delegated-admin-attribute \
    --type-name Managers \
    --attribute-type sn \
    --set "display-name:Last name"

dsconfig create-delegated-admin-attribute \
    --type-name Managers \
    --attribute-type givenName \
    --set "display-name:First Name"
```

3. To distinguish your resource type from other resources, run `dsconfig` with the `create-delegated-admin-attribute`.

   Example:

   In the following example, `Managers` are distinguished from other users using the `employeeType` attribute:

   ```
   dsconfig create-delegated-admin-attribute \
       --type-name users \
       --attribute-type employeeType \
       --set "display-name:Employee type (manager, other)" \
       --set display-order-index:4
   ```

4. To add Delegated Admin resource rights to your set of existing Delegated Admin rights, run `dsconfig` with the `create-delegated-admin-resource-rights` option.

   Example:

   In the following example, Delegated Admin are granted `reference` Admin Permission for the `Managers` REST resource type to allow viewing access:

```
dsconfig create-delegated-admin-resource-rights \
    --rights-name <existing-rights-name> \
    --rest-resource-type Managers \
    --set enabled:true \
    --set admin-permission:reference \
    --set admin-scope:all-resources-in-base
```

Example:

In the following example, Delegated Admin administrators are granted `read` and `update` Admin Permissions for `Managers` REST resource type to allow edit access:

```
dsconfig create-delegated-admin-resource-rights \
    --rights-name DArights \
    --rest-resource-type Managers \
    --set enabled:true \
    --set admin-permission:read \
    --set admin-permission:update \
    --set admin-scope:all-resources-in-base
```

Next steps

After you have created and configured a new REST resource type, add a reference Delegated Admin attribute. For more information, see step 2 in Setting up a DN reference attribute.

Differentiating resource types within the same subtree

If you have resource types with the same object class and whose subtrees overlap, you can differentiate them through include filters.

About this task

An entry is categorized as a given resource type only if the entry's attributes match the resource type's include filters.

You can only create a new resource 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 user and group resource profiles show a summary.

About this task

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.

If you configure any attributes to appear in the summary, the default selection does not appear.

To configure an attribute to appear in the summary, either use `dsconfig` to set the `include-in-summary` property or use the PingDirectory admin console to select the Include In Summary checkbox.

Steps

- For a given resource, configure each attribute to include in the summary.

Choose from:

- Use `dsconfig`.

  Use the `set-delegated-admin-attribute-prop` subcommand, where *<name_of_type>* is the resource type and *<type_of_attribute>* is the name of an existing attribute for the given resource.

  ```
  $ bin/dsconfig set-delegated-admin-attribute-prop \
    --type-name  <name_of_type>  \
    --attribute-type  <type_of_attribute>  \
    --set include-in-summary:true
  ```

  The following example includes the `cn` attribute in the summary for the `users` resource type:

  ```
  $ bin/dsconfig set-delegated-admin-attribute-prop \
    --type-name users \
    --attribute-type cn \
    --set include-in-summary:true
  ```

  > **Note**
  >
  > To remove an attribute from the summary, use the same command but change `true` to `false`.

- Use the PingDirectory admin console.

  1. Sign on to the PingDirectory admin console.
  2. From the Configuration page, click REST Resource Types.
  3. To edit a resource type, select it from the Name column.
  4. Go to the Delegated Admin Attributes section.
  5. To edit an attribute, click the Pencil icon.
  6. Select the Include In Summary checkbox.

     > **Note**
     >
     > To remove an attribute from the summary, clear its **Include In Summary** checkbox.

Customizing UI form fields

You can customize the form field for a given attribute.

About this task

Although you can customize the presentation completely, Delegated Admin provides example files to demonstrate the functionality and capabilities.

> **Note**
>
> Because the example files most likely require modifications to address your specific needs, you should have 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:

Steps

1. Go to `PingDirectory/webapps/delegator/app/customAttributes`.
2. Use a text editor to open the relevant example HTML file, as shown in 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, see the comments in `PingDirectory/webapps/delegator/app/assets/iframe-v1.js`.

4. Save the example file as a new file named `attributeName.html`.

   Example:

   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.

   Example:

   For example, because a `users` type exists with the endpoint `usersEndpoint`, you might place `cn.html` in a `customAttributes/usersEndpoint/` directory.

When locating the appropriate HTML file, Delegated Admin searches for the attribute name under the endpoint folder if one exists, and then searches under the `customAttributes` folder.

6. In the PingDirectory server, set the `attribute-presentation` field for the `Delegated Admin` attribute to `custom`.

   Example:

   ```
   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, 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 doesn't display the updated HTML file, try the following:

- Clear the browser's cache.
- Use a private browsing window.
- Try a different browser.

Setting up email invitations for a new user

You can set up email invitations for new users.

About this task

To set up email invitations for a new user:

Steps

1. Set up PingFederate for local identity profile management.

   Learn more in Configuring user self-service.

   When you complete this task, the PingFederate configuration has a local identity profile.

2. Configure Delegated Admin profile management by users.

   Learn more in Configuring user self-service.

   When you complete this task, users whom the Delegated Admin creates have the `pf-connected-identities` auxiliary object class and a `pf-connected-identity` attribute value, which provide integration with PingFederate's user self-service.

3. Instruct users to copy the email template to PingDirectory Server.

4. Create request criteria to match Delegated Admin user `ADD` requests.
5. Edit the provided email template and insert the URL to the PingFederate self-service profile management endpoint.
6. Create an SMTP external server.
7. Create a multi-part email account status notification handler for Delegated Admin user `ADD` requests.

Editing and copying the email template to the PingDirectory server

You can edit and copy the email template to the PingDirectory server using the example email template provided.

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 sign on to PingFederate and change their password and profile information.

Steps

1. Edit the template:
    1. Uncomment the line that sets the value for `profile_management_url`.
    2. 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 the PingDirectory server.

    By default, the email is sent to the address within the user's Lightweight Directory Access Protocol (LDAP) `mail` attribute.

    > **Note**
    >
    > You must provide a mail value for each user. For more information, see `common-header-fields.vm` in the email templates folder.

Next steps

For more information about the email format and further customization, see the `README` file in the templates folder.

Creating request criteria to match Delegated Admin user ADD requests

You can create request criteria to match Delegated Admin user ADD requests.

Steps

- For each user resource type for which new user email invites will be sent, create simple request criteria to match the parent distinguished name (DN) and object classes for the resource type.

    > **Note**
    >
    > The setup script includes a request criteria for the user resource type that it creates.

Example:

```
$ 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 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

You can create an SMTP external server to send emails.

About this task

To send emails:

Steps

- Configure a PingDirectory server with an SMTP server in the global configuration.

  Example:

```
$ 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

You must set an Email Account Status notification handler in the password policy in force for new users. This handler is typically the default password policy.

About this task

The notification handler references the email template in the `config/account-status-notification-email-templates` folder.

> **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.

Steps

1. Create or enable the handler:

Choose from:

- To create the handler from scratch, use the `dsconfig create-account-status-notification-handler` command.

```
$ 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.

```
$ 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.

Example:

```
$ 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

When a REST resource type is set to create new entries using a relative distinguished name (RDN) attribute other than `entryUUID`, referential integrity issues can result when the entry's RDN attribute is edited.

About this task

To preserve the referential integrity of group memberships in such a scenario, enable the referential identity plugin.

Steps

- Enable the referential integrity plugin with the following command.

```
bin/dsconfig set-plugin-prop --plugin-name "Referential Integrity" \
  --set enabled:true
```

Result

After you enable the plugin, group memberships are preserved.

Enabling log tracing

You can enable log tracing for token processing, HTTP request and response actions, and API debugging.

Steps

- To view OAuth token processing and full HTTP request and response tracing, enable the debug trace logger with the following command.

```
$ 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
```

Specifying a custom hostname and port for your PingDirectory server

If an application is installed on a separate server from your PingDirectory server instance, you must specify where the PingDirectory server instance is found within the Delegated Admin application's `config.js` file.

The `DS_HOST` configuration variable should match your PingDirectory server's hostname. The `DS_PORT` configuration variable should match your PingDirectory server's HTTPS port.

> **Note**
>
> You don't need to specify these variables if you're hosting the Delegated Admin application alongside your PingDirectory server instance, as described in previous sections.

Changing the application logo

You can change the Ping Identity logo to your corporate logo.

About this task

Support for corporate logos includes, but is not limited to, the following file types:

- `JPG`
- `PNG`
- `SVG`

To maintain an appropriate aspect ratio, logo images are resized in Delegated Admin to a height of 22px and a maximum width of 150px.

Steps

1. To change the Ping Identity logo to your corporate logo, add the following line to the `config.js` configuration file:

```
window.HEADER_BAR_LOGO = '<filename>';
```

2. Add the logo to the build directory.

> **Note**
>
> Make sure to use the same file name specified in step 1.

Result

The corporate logo appears in the header, and the Ping Identity logo relegates to the sidebar in grayscale, preceded by "Powered by".

Configuring the session timeout

By default, Delegated Admin has an idle session timeout value of 30 minutes. To adjust this value, perform the following steps.

Steps

1. Open the `config.js` configuration file in a text editor.
2. Add the following line to the configuration file:

```
window.TIMEOUT_LENGTH_MINS=<TimeoutValue>;
```

<*TimeoutValue*> is an integer that represents the session timeout value in minutes.

To view an example outline that features this setting, see `example.config.js`.

3. Save your changes to `config.js`.

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.

About this task

In the Delegated Admin GUI, the left navigation pane provides a Reporting tab under the top-level sections, such as Users and Groups.

Administrators can also use the Reporting page to upload users and group members in bulk with a `.csv` file. The `.csv` file should contain headers with the exact attribute names of the users resource type. Users can be uploaded with missing info for some of the attributes but the required attributes must be present and specified. For a group administrator, uploading users will also add them to the appropriate group. Click `Upload` on the Reporting page and select a `.csv` file to upload.

On each Reporting tab, you can:

- Filter the columns or attributes to display.
- Filter the users returned, such as all users with the first name John. Only exact matches are returned.
- Sort columns by clicking the column name.

Steps

- 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 the PingDirectory server required to access the full functionality provided by each release of Delegated Admin.

> **Note**
>
> All versions of Delegated Admin require PingFederate 9.0.0 or later.

Delegated Admin	PingDirectory server
5.0.0	9.3.0.0
4.10.0	9.1.0.0
4.9.0	9.1.0.0 EA
4.8.0	9.0.0.0
4.7.0	9.0.0.0 EA
4.6.0	8.3.0.0
4.5.0	8.3.0.0 EA
4.4.1	8.2.0.0
4.4.0	8.2.0.0
4.3.0	8.2.0.0 EA

Configuring the PingFederate server

PingFederate offers many configuration options. This section provides an example PingFederate configuration that supports Delegated Admin.

> **Note**
>
> The PingFederate information in this section is based on the PingFederate 10.1 interface.

Configuring the PingFederate server includes the following:

- Configuring PingFederate as the identity provider
- Configuring the OAuth server
- Configuring the PingDirectory server as the token validator (create OAuth client for PingDirectory)
- Configuring Delegated Admin as a new client (create OAuth client for Delegated Admin)
- Setting cross-origin resource sharing (CORS) settings

Configuring PingFederate as the identity provider

The following task configures the PingFederate server as the identity provider (IdP) for the PingDirectory server.

Before you begin

Download the LDAPS certificate from the PingDirectory server. For more information, see Exporting certificates.

Steps

1. Sign on to the PingFederate admin console.
2. Import the PingDirectory server LDAPS certificate:
    1. Go to Security → Certificate & Key Management → Trusted CAs.
    2. Click Import, click Choose File to browse to the certificate, click Next, and then click Save.
3. Add an Lightweight Directory Access Protocol (LDAP) datastore:
    1. Go to System → Data Stores.
    2. Click Add New Data Store.
    3. Specify a Name for the data store.
    4. Set Type to Directory (LDAP).
    5. Click Next.
    6. In the Hostname(s) field, enter the PingDirectory server host name and LDAPS port, separated by a colon (for example, 10.101.113.75:1636) and click Add.
    7. Select the Use LDAPS check box.
    8. Set LDAP Type to PingDirectory.
    9. In the User DN field, enter one of the following values based on your PingDirectory configuration:
        - `cn=dmanager`
        - `cn=Directory Manager`

        > **Note**
        >
        > These values are based on the assumption that Delegated Admin will run as the directory manager.

    10. In the Password field, specify the root password.
    11. 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.
    12. Click Test Connection.
    13. Click Next.
    14. Click Save.
4. Create the HTML form IdP Adapter.

The adapter authenticates users against the PingDirectory server.

1. Go to Authentication → IdP Adapters → Create New Instance
2. In the Instance Name field, enter a name such as `PingDirectoryIdP`.
3. Specify an Instance ID.
4. Set Type to HTML Form IdP Adapter.
5. Click Next.
6. Go to the bottom of the page and click Manage Password Credential Validators.
7. Create a validator to authenticate users against the 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`.

        > **Note**
        >
        > These values are used later.

    10. Click Next, Done, or Save until you reach the Create Adapter Instance screen.
8. Add a new row to Password Credential Validators, choose the new LDAP Password Credential Validator, and click Update.
9. Go to the Extended Contract tab and extend the adapter contract with `entryUUID` and `cn`.
10. Go to the Adapter Attributes tab, select `entryUUID` for a pseudonym, and then click Next, Next, Done, and Save.

    Learn more about Configuring the LDAP Username Password Credential Validator in the PingFederate documentation.

5. Enable session tracking:
    1. Go to Authentication → Sessions

2. Select the Track Adapter Sessions For Logout check box.
3. Select the Track Revoked Sessions On Logout check box.
4. Select the Enable Authentication Sessions For All Sources check box.
5. Click Save.

Configuring the OAuth server

The following task configures the PingFederate server for OAuth and OpenID Connect (OIDC) authentication.

Steps

1. Sign on to the PingFederate admin console.
2. Set the identity provider (IdP) adapter mapping:
    1. Go to Authentication > OAuth > IdP Adapter Grant Mapping.
    2. From the Source Adapter Instance list, select the IdP adapter you created in Configuring PingFederate as the identity provider and click Add Mapping.
    3. Click Next.

    > **Note**
    > No attribute source is needed.

    4. 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

    5. Click Next and then click Next again.
    6. Click Save.
3. Set up Access Token Management.

    Select an existing instance or click Applications > OAuth > Access Token Management > Create New Instance.

    Choose from:

    - If selecting an existing instance, click the Instance Configuration tab.

    > **Note**
    > With an existing instance, a JSON Web Token (JWT) is configured automatically.

    - If creating a new instance, specify the required fields and set Type to JSON Web Tokens.

> **Note**
>
> Take note of your new instance name. You'll need that information later.

1. Use symmetric encryption for the JWT by adding a row in the Symmetric Keys section using 32 bytes or 64 characters of hex.

   > **Note**
   >
   > 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.

2. Set JWS Algorithm to HMAC Using SHA-256.
3. Set Active Symmetric Key ID to your symmetric key and click Next.
4. On the Session Validation tab, select all options and click Next.
5. 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:
   1. Go to Applications > OAuth > Access Token Mappings.
   2. Set Context to Default, set Access Token Manager to the access token manager you created in the last step, and click Add Mapping.
   3. Click Next in the Attribute Source & User Lookup section to go to the Contract Fulfillment section.
   4. In the sub row, make the following selections:
      - In the Source list, select Persistent Grant.
      - In the Value list, select USER_KEY.
   5. Click Next until you reach the Summary section. Click Save.
5. Set up the OpenID Connect policy:
   1. Go to Applications > OAuth > OpenID Connect Policy Management.
   2. Click Add Policy.
   3. Specify a Policy ID.
   4. Specify a Name.
   5. Choose the previously created access token manager and click Next.
   6. Delete all extended contract attributes except `sub`.

      Other scopes are defined, if configured.
   7. Click Next to reach the Contract Fulfillment section.
   8. Fulfill the OpenID Connect (OIDC) contract sub with the access token attribute `sub`.

9. Click Next and then click Done.
10. 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:
    1. Go to System > OAuth Settings > Scope Management.
    2. Click the Exclusive Scopes tab.
    3. Add a scope with a Scope Value of `urn:pingidentity:directory-delegated-admin` and a Scope Description of `DAScope`.
    4. Click Save.

Configuring the PingDirectory server as the token validator (create OAuth client for PingDirectory)

When creating a PingFederate access token validator in the PingDirectory server, use the `pingdirectory` client ID and secret.

About this task

The PingDirectory server uses an identity mapper to match the `sub` claim against the `entryUUID` attribute.

To configure the PingDirectory server as the token validator:

Steps

1. Sign on to the PingFederate admin 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 enter 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 the Default Access Token Manager to the access token manager you created in step 3 of Configuring the OAuth server.
10. Click Save.

Configuring Delegated Admin as a new client (create OAuth client for Delegated Admin)

The following task configures Delegated Admin as a new client and outlines how to create an OAuth client for Delegated Admin.

About this task

To configure Delegated Admin as a new client:

Steps

1. Sign on to the PingFederate admin 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.

> **Important**
>
> Do not set a client secret.

6. For Redirect URIS, enter the appropriate URI for your environment based on the following table and then click Add.

For Delegated Admin on a PingDirectory server or a PingDirectoryProxy server	`https://<server-host>:<server-port>/delegator/*`
For Delegated Admin on a web server hosted locally	`http://localhost:<server-port>/*`

7. Make the following selections:
    1. In the Bypass Authorization Approval section, select Bypass.
    2. In the Exclusive Scopes section, select Allow Exclusive Scopes and then select urn:pingidentity:directory-delegated-admin.
    3. In the Allowed Grant Types section, select Authorization Code.
    4. In the Default Access Token Manager list, select the token manager that you created in step 3 of Configuring the OAuth server.
    5. Select the check box for Require Proof Key for Code Exchange (PKCE).
    6. In the OpenID Connect section, select the OpenID Connect (OIDC) policy that you created in step 5 of Configuring the OAuth server.
8. Click Save.

Next steps

After completing the previous steps, configure the following settings to display the name of the administrator who is signed on to the client application:

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 supplied to the Delegated Admin application.

3. Set the `PROFILE_SCOPE_ENABLED` configuration variable for Delegated Admin in the `config.js` file to `true`.

```
/**
 * 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,
};
```

Setting cross-origin resource sharing (CORS) settings

The CORS settings can be set through the PingFederate admin console.

Steps

1. Sign on to the PingFederate admin 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 Allowed Origin, using the host name and HTTPS listener port for the PingDirectory server.

> **Important**
>
> Don't include `/delegator` in the **Allowed Origin** field. Doing so causes Delegated Admin to throw a token validation error when you attempt to sign on.

For example, consider the following table:

For Delegated Admin on a PingDirectory server or a PingDirectoryProxy server	`https://<server-host>:<server-port>`
For Delegated Admin on a web server hosted locally	`http://localhost:5000/*`

5. Click Save.

Changing the default OIDC grant type

To change the Delegated Admin application's default OpenID Connect (OIDC) grant type, use the PingFederate admin console and the Delegated Admin `config.js` file.

About this task

To improve authentication security, switch your default OIDC grant type to Authorization Code with Require Proof Key for Code Exchange (PKCE). The Authorization Code with PKCE grant type hides access tokens during authentication with JavaScript applications, in comparison to the Implicit grant type that displays access tokens in the URL redirect during OIDC authentication.

> **Note**
>
> For more information, see OAuth Grant Types.

The following example changes the default OIDC grant type from Implicit to Authorization Code with Require Proof Key for Code Exchange (PKCE).

Steps

1. In the PingFederate admin console, go to Applications > OAuth > Clients.
2. From the Clients list, select the dadmin client.
3. In the Allowed Grant Types section:
    1. Select the Authorization Code check box.
    2. Clear the Implicit check box.
    3. Select the Require Proof Key for Code Exchange (PKCE) check box.
    4. Click Save.
4. From your `<server-root>` directory, open the Delegated Admin application's `config.js` file and set the `AUTHENTICATE_WITH_PKCE` variable to `true`.

    Example:

    ```
    /*
     * Indicates if this app should authenticate using the 'Authorization Code with PKCE' OAuth grant.
     * If true, the 'Authorization Code with PKCE grant will be used. If false, the 'Implicit' grant
     * will be used. * DEFAULT: window.AUTHENTICATE_WITH_PKCE = true;
     */
    window.AUTHENTICATE_WITH_PKCE = true;
    ```

    > **Note**
    >
    > If you don't already have the `AUTHENTICATE_WITH_PKCE` variable in your `config.js` file, you must add it.

PingDataSync Server Administration Guide

The PingDataSync server is an efficient, Java-based server that provides high throughput, low latency, and bidirectional real-time synchronization between two endpoint topologies, such as PingDirectory servers, PingDirectoryProxy servers, PingOne, or relational database management systems (RDBMS).

PingDirectory product documentation

Support

https://support.pingidentity.com

Overview of the PingDataSync server process

The following topics present a general overview of the PingDataSync process and examples for use:

Data synchronization process

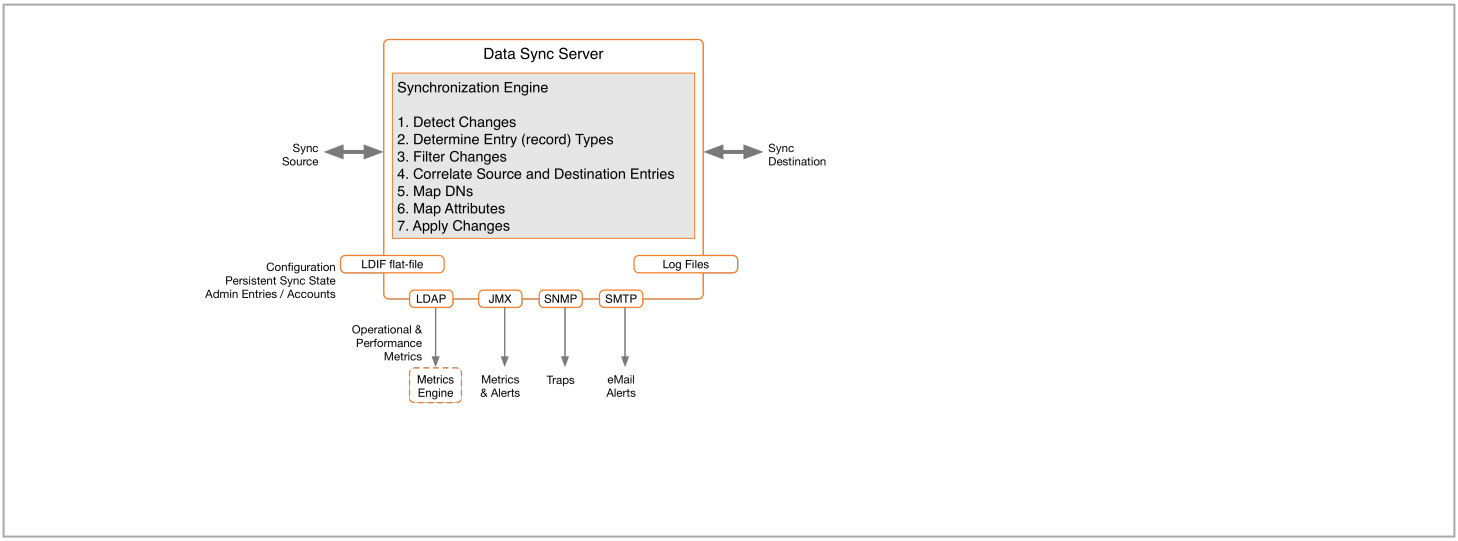
PingDataSync performs point-to-point synchronization between a source endpoint and a destination endpoint. An endpoint is defined as any source or destination topology of directory or database servers.

PingDataSync synchronizes data in one direction or bidirectionally between endpoints. For example, in a migration phase from Sun Directory server to a 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 also contains no single point of failure, either for detecting changes or for applying changes. PingDataSync 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, because they each verify the full entry before making changes.

Synchronization architecture

PingDataSync uses a virtualized, dataless approach that does not store directory data locally. The log files, administrator entries, configuration, and sync state information are stored as flat files (LDIF format) within the system. No additional database is required.



Change tracking, monitoring, and logging

PingDataSync 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 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 PingDirectory server topologies, PingDataSync 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 uses the server's Retro Change Log, which provides a detailed summary of each change.

- For Active Directory (AD), PingDataSync uses the DirSync control, which polls for object attribute changes.
- For RDBMS systems, PingDataSync uses a Ping Identity Server SDK plugin to interface with a customized RDBMS change log table. The database triggers on each table record all INSERT, UPDATE, and DELETE operations to the change log table.

Monitoring, Alerts, and Alarms

PingDataSync 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 provides standard logs (`sync`, `access`, `error`, `failed-operations`, `config-audit.log`, and `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 runs as a standalone Java process with two synchronization modes: standard and notification.

Standard synchronization

In standard synchronization mode, PingDataSync 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.

Notification synchronization

In notification synchronization mode, PingDataSync skips the fetch and compare phases of processing, notifies the destination that a change has happened, and provides the details of the change. Notification mode is currently available for the PingDirectory and Alcatel-Lucent 8661 directory and proxy servers only.

PingDataSync operations

PingDataSync provides seamless integration between disparate systems to transform data using attribute and DN mappings. A bulk resynchronization operation can be run to 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. After 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 attribute 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 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 the 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 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.

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 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 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 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, such as 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

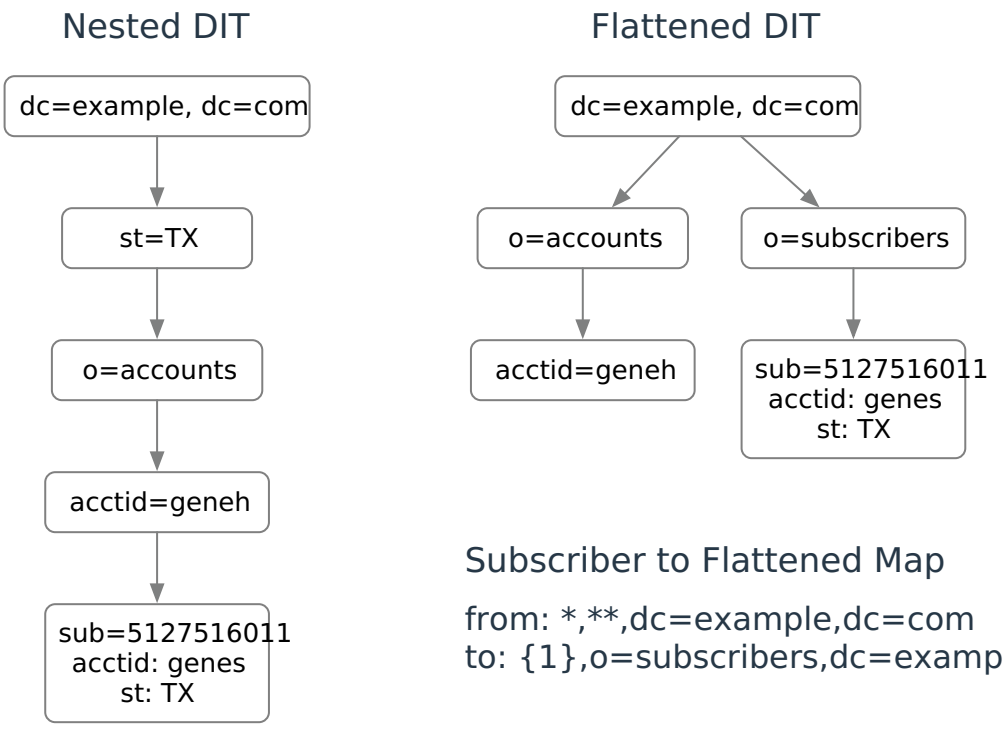
Defines mappings for use when destination DNs differ from source DNs. These mappings allow the use of wild cards for DN transformations. A single wild card ("*") matches a single RDN component and can be used any number of times. The double wild card ("**") matches zero or more RDN components and can 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 figure illustrates 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.
```

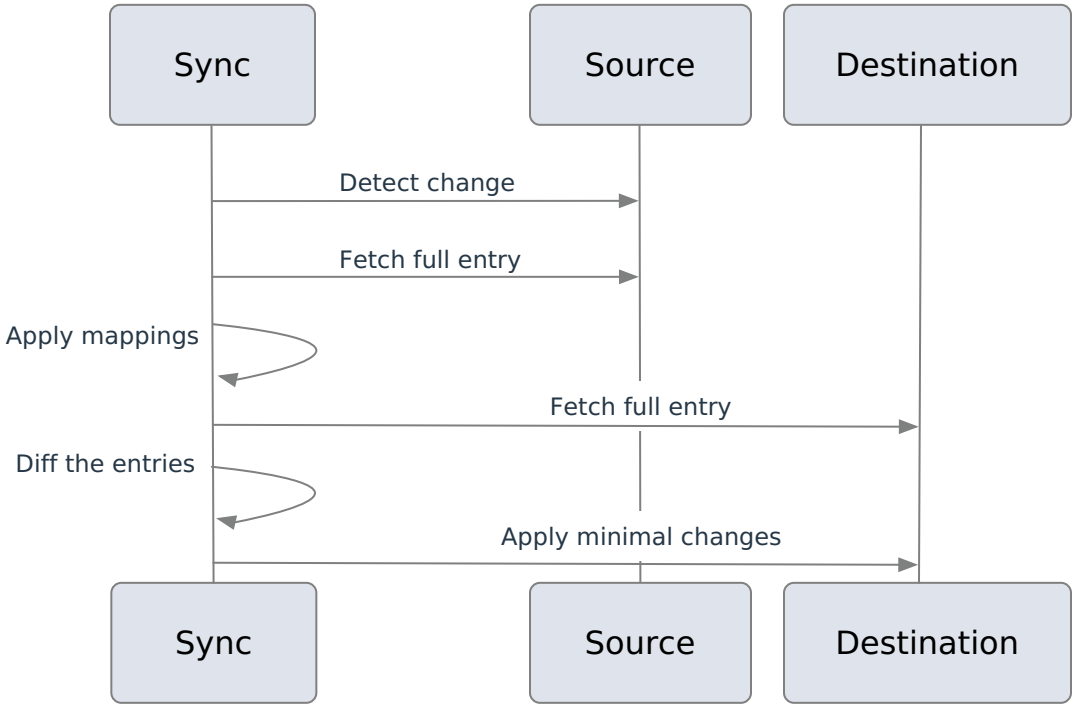
Sync flow examples

PingDataSync processes changes by fetching the most up-to-date, full entries from both sides and then comparing them. This process flow is called standard synchronization mode.

The processing flow differs depending on the type of PingDataSync 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

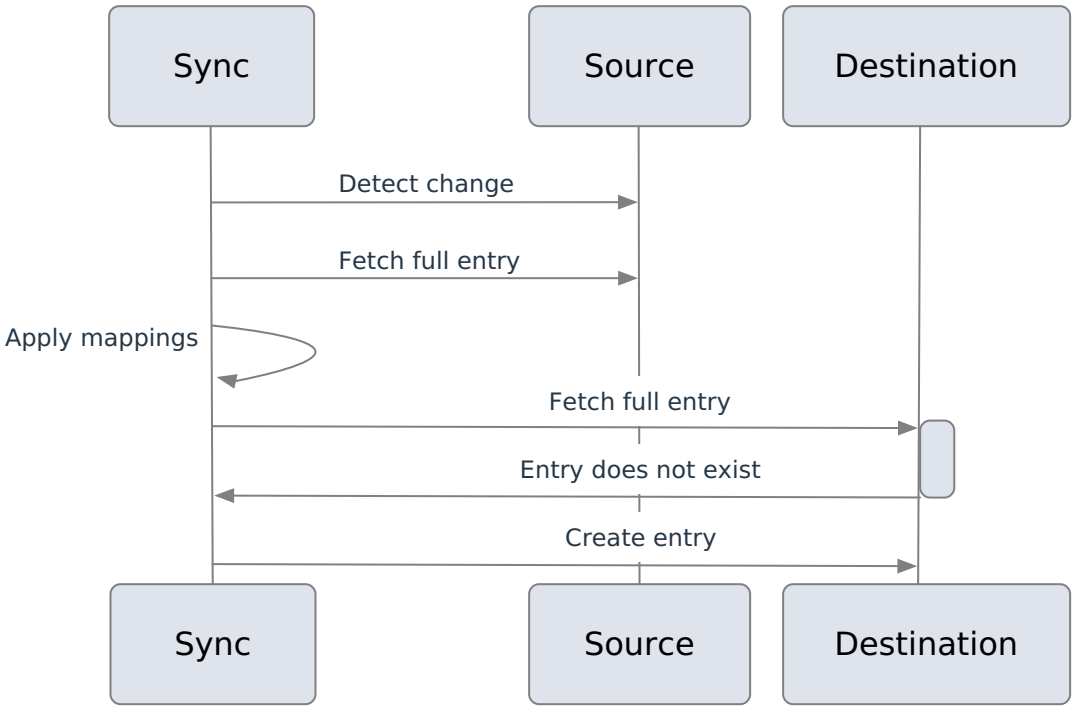


Steps

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

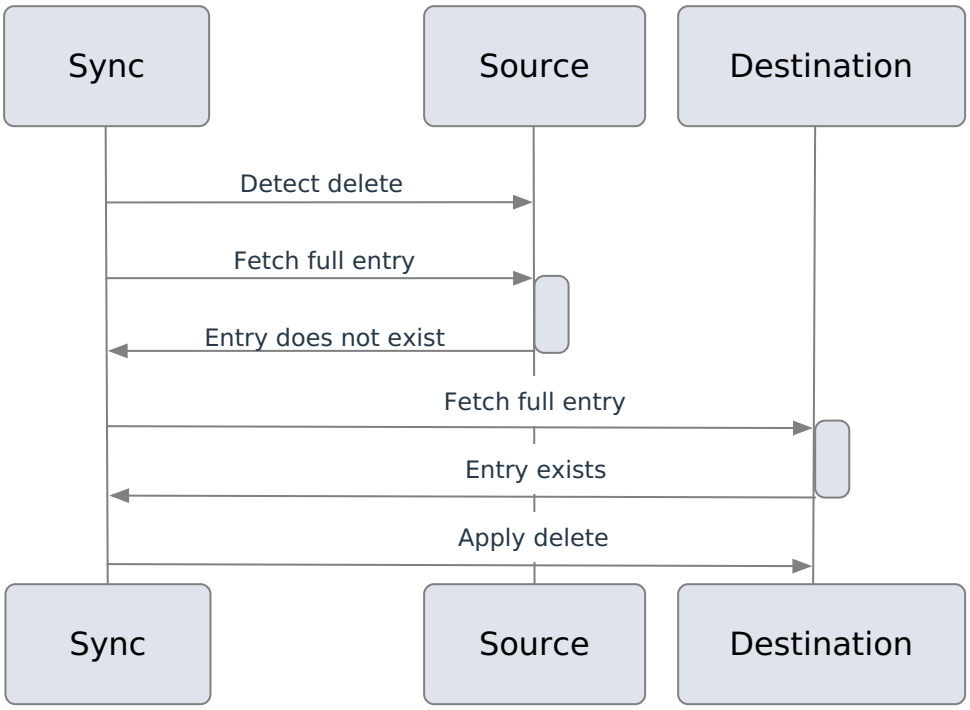


Steps

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

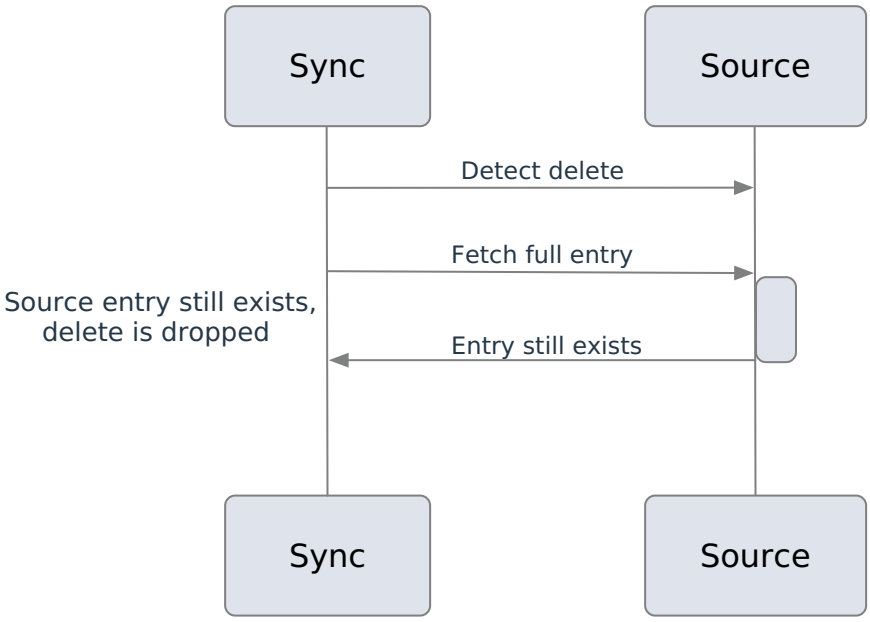


Steps

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

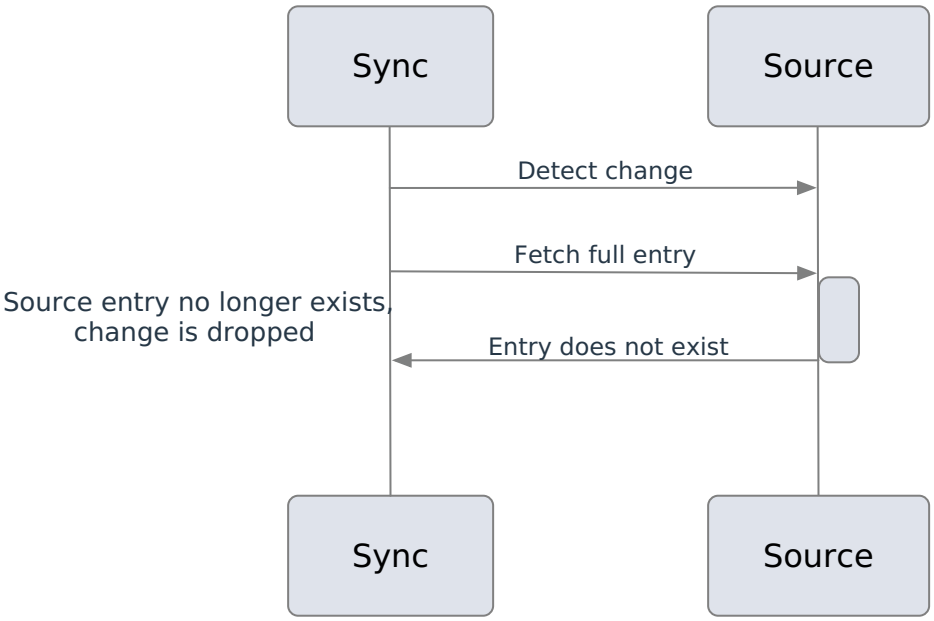


Steps

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



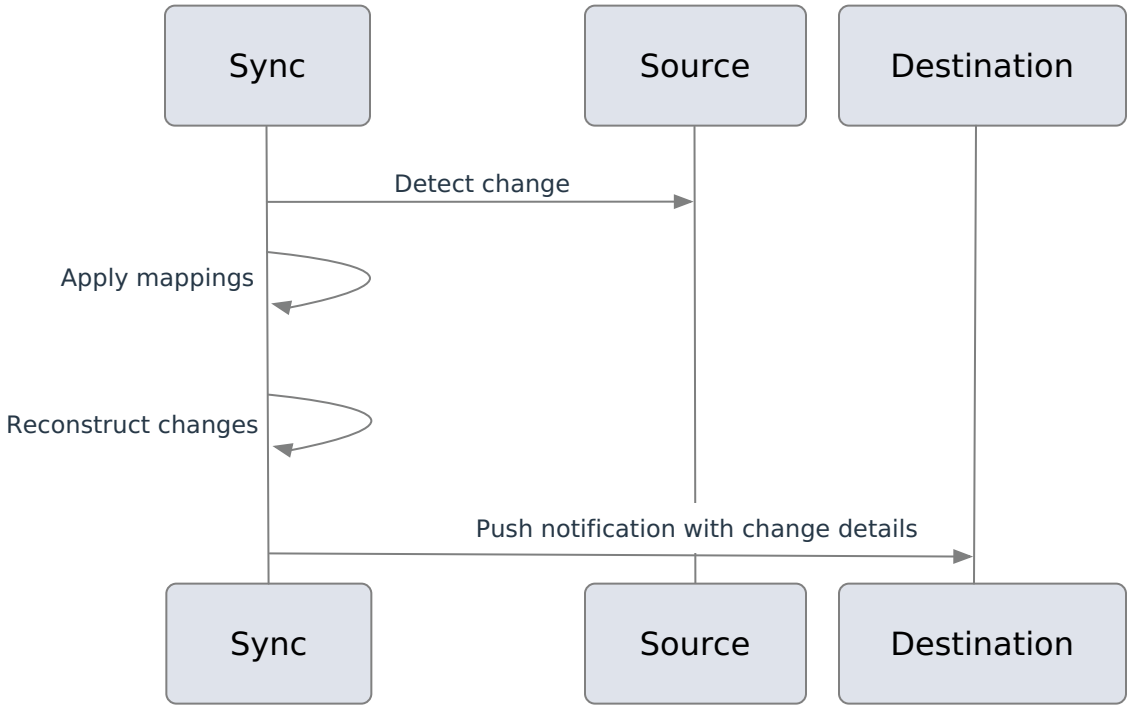
Steps

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.

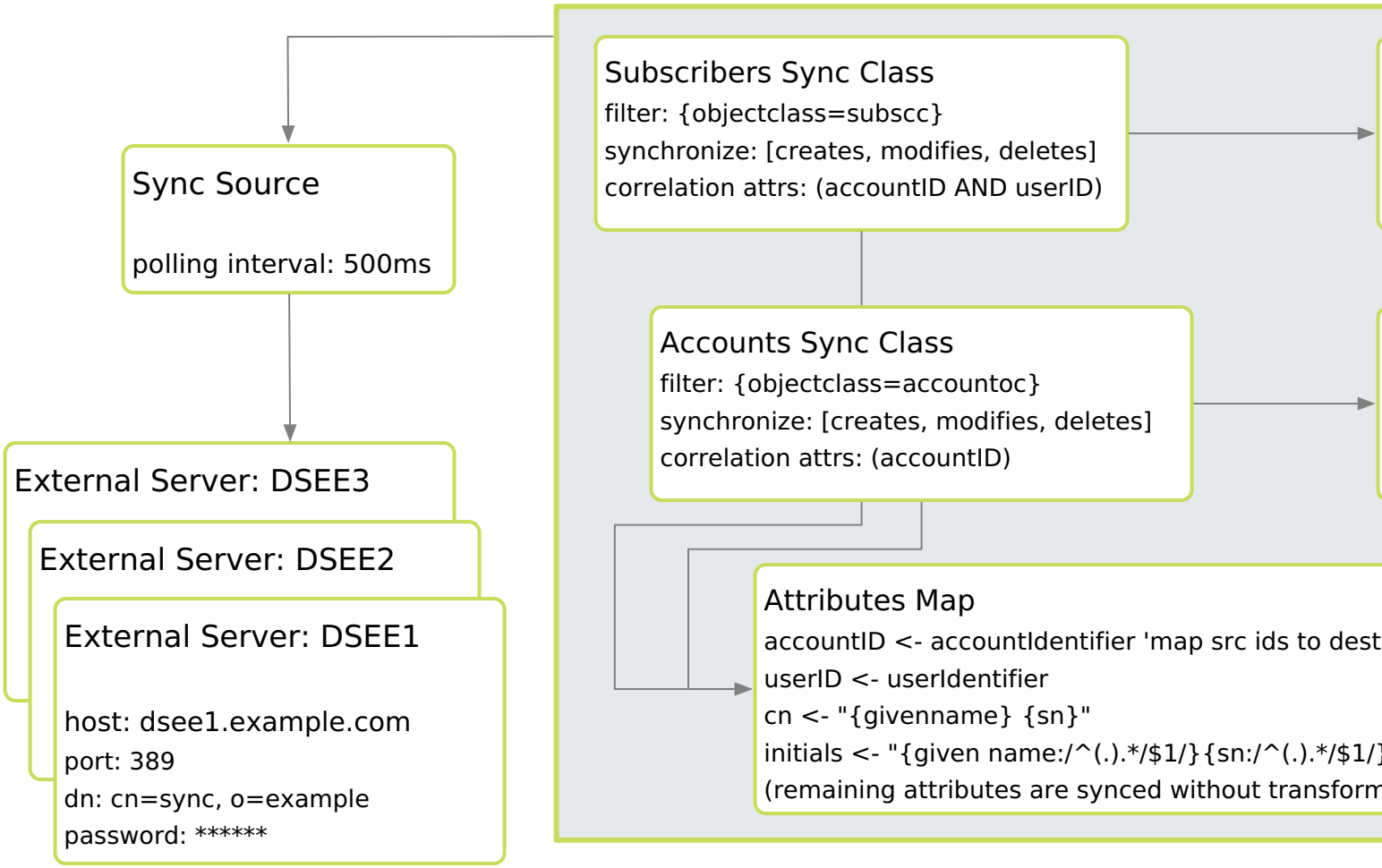
Sample synchronization

The following is a synchronization migration example from a Sun Directory Server Enterprise Edition (DSEE) topology to the 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.



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 might 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.
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.
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.

Directories/Files/Tools	Description
config	Stores the configuration files for the backends (admin, config) as well as the directories for messages, schema, tools, and updates.
logs	Stores log files.
AD-Password-Sync-Agent.zip	The Active Directory Sync Agent package.

Changing the administrative password

Users that authenticate to the Configuration API or the admin 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.

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
```

Running the server as a Microsoft Windows service

The server can run as a Windows service on Windows Server 2016, 2019, and 2022. This enables logging out of a machine without the server being stopped.

Register the service

Steps

1. Stop the server using the `bin\stop-server` command. 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 begin with `windows-service-wrapper`. They are configured to rotate each time the wrapper starts or because of file size. Only the last three log files are retained. These configurations can be changed in the `<server-root>\config\wrapper.conf` file.

Configuring the PingDataSync server

PingDataSync provides a suite of tools to configure a single server instance or a group of servers. All configuration changes to the server are recorded in the `config-audit.log`. Before configuring PingDataSync, review Configuration components.

Topics include:

- Configuration checklist
- Sync user account
- Configure PingDataSync in standard mode
- Using the Configuration API
- Configuration with the `dsconfig` tool

- Topology configuration
- Domain Name Service (DNS) caching
- IP address reverse name lookups
- Configure the synchronization environment with `dsconfig`
- Prepare external server communication
- HTTP connection handlers
- The `resync` command
- The `realtime-sync` tool
- Configure the PingDirectory server backend for synchronizing deletes
- Configure DN maps
- Configure synchronization with JSON attribute values
- Configure fractional replication
- Configure failover behavior
- Configure traffic through a load balancer
- Configure authentication with a SASL external certificate
- Configure an LDAPv3 Sync Source
- Server SDK extensions

Configuration checklist

Before any deployment, determine the configuration parameters necessary for the synchronization topology.

For a better configuration experience, gather the following information before you start your configuration.

External servers

> **Warning**
>
> As a best practice, external server definitions should not point at load balancers. Instead, direct connections should be made to the required systems.

External server type

Determine the type of external servers included in the synchronization topology. Supported servers include:

- **PingDirectory server**
- **PingDirectoryProxy server**
- **Sun Directory Server Enterprise Edition**

- Sun Directory Server
- Oracle Unified Directory
- OpenDJ
- Microsoft Active Directory (AD)
- Generic LDAP directories

LDAP connection settings

Determine the following for each external server instance included in the synchronization topology:

- Host
- Port
- Bind distinguished name (DN)
- Bind password

Security and authentication settings

Determine the following:

- The secure connection types for each external server, such as SSL or StartTLS
- The authentication methods for external servers, such as simple authentication or external SASL mechanisms

> **Important**
>
> If you are synchronizing encoded passwords or clear-text passwords in particular, the connection should be secure. To synchronize to or from a Microsoft AD system, establish an SSL or StartTLS connection to PingDataSync. You should also enable password encryption for synchronization from AD or when synchronizing clear-text passwords. For more information, see Configuring password encryption.

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. You can define a prioritized list of external servers for failover purposes.

`Sync Destination`

Determine which external server is the `Sync Destination` for the synchronization topology. You can define a prioritized list of external servers 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 you are 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 which operations to synchronize:

- `create`
- `modify`
- `delete`

DNs

Determine the differences between the DNs from the `Sync Source` topology to the `Sync Destination` topology. For example, if there are structural differences in each directory information tree (DIT).

Destination correlation attributes

Determine the correlation attributes used to associate a source entry to a destination entry during synchronization. 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 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 that several destination searches with different combinations of attributes can be performed until an entry matches. For LDAP server endpoints, use the DN to correlate entries.

For example, to correlate entries in LDAP deployments, specify the attribute lists `dn,uid`, `uid,employeeNumber` and `cn,employeeNumber`. PingDataSync searches for a corresponding entry that has the same `dn` and `uid` values. If the search fails, it then searches for `uid` and `employeeNumber`. If that search fails, it searches for `cn` and `employeeNumber`. If none of these searches are successful, the synchronization change is aborted and a message is logged.

> **Note**
>
> 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 are 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 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`.

Attribute consideration	Description
Attribute mappings	Determine how attributes are mapped from the `Sync Source` to the `Sync Destination`. For example, if they're mapped directly, mapped based on attribute values, or mapped based on attributes that store DN values.
Automatically mapped source attributes	Determine if there are attributes that can be automatically synchronized with the same name at the `Sync Source` to `Sync Destination`. For example, if you can set direct mappings for `cn`, `uid`, `telephoneNumber`, or for all attributes.
Non-auto mapped source attributes	Determine if there are attributes that shouldn't be automatically mapped. For example, the `Sync Source` might have an attribute, `employee`, while the `Sync Destination` has a corresponding attribute, `employeeNumber`. If an attribute isn't automatically mapped, you must provide a map to synchronize it.
Conditional value mapping	Determine if some mappings should only be applied some of the time as a function of the source attributes. **Note** You can use conditional value mappings 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 multivalued. 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 isn't an ordinary mapping, the attribute is not created on the destination.

Sync user account

PingDataSync creates a Sync User account DN on each external server. The account (by default, `cn=Sync User`) is used exclusively by PingDataSync to communicate with external servers. The entry is important in that it contains the credentials (DN and password) used by PingDataSync to access the source and target servers. The Sync User account resides in different entries depending on the targeted system:

- For the PingDirectory server and PingDirectoryProxy servers, the Sync User account resides in the configuration entry (`cn=Sync User,cn=Root DNs,cn=config`).
- For Sun Directory Server, Sun DSEE, OpenDJ, Oracle Unified Directory, and generic LDAP directory topologies, the Sync User account resides under the base DN in the `userRoot` backend (`cn=Sync User,dc=example,dc=com`). The Sync User account should not reside in the `cn=config` branch for Sun Directory Server and DSEE machines.
- For Microsoft Active Directory (AD) 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 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 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.

`xref:pd_sync_use_create_sync_pipe_tool.adoc[] 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 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.

Using the create-sync-pipe tool to configure synchronization

You can use the `create-sync-pipe-config` utility to configure a Sync Pipe. After the configuration is completed, you can adjust settings using the `dsconfig` tool.

About this task

> **Important**
>
> 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 PingDirectory server destination, you must allow pre-encoded passwords in the default password policy. You must also configure password encryption on the destination. Make 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**
>
> You can set the `require-secure-connection-for-clear-text-passwords` property to `false` when working in a test environment.
> If the `password-synchronization-format` property is set to `clear-text`, and the `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.

To configure PingDataSync with the `create-sync-pipe-config` command:

Steps

1. Start PingDataSync:

   ```
   $  <server-root>/bin/start-server
   ```

2. From the `bin` directory, run the `create-sync-pipe-config` tool:

   ```
   $bin/create-sync-pipe-config
   ```

3. In the Initial Synchronization Configuration Tool menu, press Enter (yes) to continue the configuration.
4. In the Synchronization Mode menu, press Enter to select Standard Mode.
5. In the Synchronization Directory menu, select oneway(1) or bidirectional(2) for the synchronization topology.

   This example assumes bidirectional synchronization.

6. In the Source Endpoint Type menu, select the directory or database server for the first endpoint.
7. In the Source Endpoint Name menu, enter a name for the endpoint server, or press Enter to accept the default.
8. In the Base DNs menu, enter the base distinguished name (DN) on the first endpoint topology where the entries will be searched.

   For example, `dc=example,dc=com`.

9. Select an option for the server security.

10. Enter 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.

    Result:

    You can now configure the servers in the destination endpoint topology.

14. 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.

    After you configure the source and destination topologies, PingDataSync prepares each external server by testing the connection to each server. Step 15 determines if each account has the necessary privileges (root privileges are required) to communicate with and transfer data to each endpoint during synchronization.

16. In the Sync Pipe Name menu, create a name for the Sync Pipe, 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.

17. In the SyncClass Definitions menu, enter `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.

18. Enter a name for the new Sync Class, such as `server1_to_server2` .

19. In the Base DNs for Sync Class menu, enter one or more base DNs to synchronize specific subtrees of a directory information tree (DIT).

    > **Note**
    >
    > Entries outside of the specified base DNs are excluded from synchronization. Make sure the base DNs do not overlap.

20. In 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 are excluded from synchronization.

21. In the Synchronized Attributes for Sync Class menu, specify which user 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. Operational attributes are not included in any attribute mapping.

22. In the Operations for Sync Class menu, select the operations to synchronize for the Sync Class, or press Enter to accept the default (`1,2,3`).

23. 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`.
24. In 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.
25. 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.
26. 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`).

    You can reuse the batch file to set up other topologies.

Next steps

Apply the configuration changes to the local PingDataSync instance by using a `dsconfig` batch file. Any Server SDK extensions should be saved to the `<server-root>/lib/extensions` directory.

The next step is 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. The attribute mapping options are available from the PingDataSync main menu.

Steps

1. On PingDataSync, 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
```

Configuring server locations

About this task

PingDataSync 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 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, 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 itself. In this example, set the location to `austin`.

```
$ bin/dsconfig set-global-configuration-prop \
  --set location:austin
```

Using the Configuration API

PingDataSync 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. Because of 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.

dsconfig tool">

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 can 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. See Sorting and filtering configuration objects 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`.	`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 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-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 can also specify sort parameters or paging parameters, and 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 to sort by.
`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 might 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 to determine the success or failure of a request. The following table lists 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 might 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 because of 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. 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 might 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."
}
```

dsconfig tool">

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 in the `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
```

dsconfig in interactive mode">

Using `dsconfig` in interactive mode

Running `dsconfig` in interactive command-line mode provides a menu-driven interface for accessing and configuring the PingDataSync 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.

dsconfig in non-interactive mode">

Using `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` option 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}` argument specifies which general action to perform. The following example 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 example 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 list 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.

dsconfig batch mode">

Using `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 also provides a `docs/sun-ds-compatibility.dsconfig` file for migrations from Sun/Oracle to PingDataSync machines.

Topology configuration

Topology configuration enables grouping servers and mirroring configuration changes automatically. It uses a primary/secondary architecture for mirroring shared data across the topology.

All writes and updates are forwarded to the primary, 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 primary requirements and selection

A topology primary 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 primary. The primary always sends a digest of its subtree contents on each update. If the node has a different digest than the primary, it knows it's not synchronized. The servers will pull the entire subtree from the primary if they detect that they are not synchronized. A server can detect it is not synchronized with the primary under the following conditions:

- At the end of its periodic polling interval, if a server's subtree digest differs from that of its primary, 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 primary 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 primary, 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 primary. If a new server can be the primary, it will communicate that to the other servers, if no other server has advertised that it should be the primary. This ensures that all servers accept the same primary at approximately the same time (within a few milliseconds of each other). If there is no better primary, the initial primary maintains the role.

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

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

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

Topology components

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

Server configuration settings

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

The Global Configuration object contains a single topology setting, `force-as-master-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 primary because a majority of servers is not available. A server with this setting enabled will be assigned the role of primary if no suitable primary can be determined. Learn more in Topology primary requirements and selection.

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 might warrant a new primary 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 primary is found, the polling frequency is adjusted to 100 milliseconds until a new primary 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 primary 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 primary 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 primary, if the primary is not known.
- The instance name of the current primary, or if a primary is not set, a description stating why a primary is not set.
- A flag indicating if this server thinks that it should be the primary.
- A flag indicating if this server is the current primary.
- A flag indicating if this server was forced as primary.
- 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 primary, or another node in the topology if the primary is unknown.
- The amount of time in milliseconds where multiple primaries were detected by this server.
- The amount of time in milliseconds where no suitable server is found to act as primary.
- 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 primary:

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

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 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.

To reduce delays because of 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` 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 might 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)

dsconfig">

Configure the synchronization environment with `dsconfig`

The `dsconfig` tool can be used to configure any part of PingDataSync, but will likely be used for more fine-grained adjustments. If configuring a Sync Pipe for the first time, use the `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, run the following command:

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

Updates to servers in the group are made using the `--applyChangeTo server-group` 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 servers.

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 can occur between PingDataSync 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 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 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 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 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. 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

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 before 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 might be distinct servers residing on distinct hosts and locations. Tracing the request flow on distributed systems can be challenging, because 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 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 can 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 PingAuthorize 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");
```

Configuring 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 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.

```
$ 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"'
                        {pingdir}/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"'
                                {pingdir}/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='{pingdir}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='{pingdir}-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='{pingdir}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="dscfg- password-attribute" resultCode=0
                               resultCodeName="Success" etime=0.065 preAuthZUsedPrivileges="bypassacl,config-read"
entriesReturned=1
```

Configuring the PingDataSync server to use an HTTP proxy server

Some organizations configure their network so that internal systems cannot directly access the Internet, but instead must use an HTTP proxy server to access external services.

The PingDataSync server now supports using an HTTP proxy server in conjunction with the following components:

- The Amazon Key Management Service cipher stream provider
- The Amazon Secrets Manager cipher stream provider
- The Amazon Secrets Manager passphrase provider
- The Amazon Secrets Manager password storage scheme
- The Azure Key Vault cipher stream provider
- The Azure Key Vault passphrase provider

- The Azure Key Vault password storage scheme
- The PingOne pass-through authentication plugin
- The PingOne sync source and destination
- The Pwned Passwords password validator
- The SCIMv1 sync destination
- The SCIMv2 sync destination
- The Twilio alert handler
- The Twilio OTP delivery mechanism
- The UNBOUNDID-YUBIKEY-OTP SASL mechanism handler

Setting up the server to use an HTTP proxy server involves two steps, which are described in the following sections.

1. Creating an HTTP proxy external server in the configuration.
2. Configuring the appropriate component(s) to use the HTTP proxy server.

Creating an HTTP proxy external server

An HTTP proxy external server configuration object provides information about a proxy server that should be used. At present, we only support HTTP proxy servers (which can handle both unencrypted HTTP and encrypted HTTPS connections), with or without authentication.

HTTP proxy external server definitions support the following configuration properties:

`server-host-name`

The resolvable name or IP address of the HTTP proxy server to use. This is required.

`server-port`

The port on which the HTTP proxy server is listening for connections. This is required.

`basic-authentication-username`

The username to use if the proxy server requires authentication. This should be omitted if the proxy server does not require authentication.

`basic-authentication-passphrase-provider`

The passphrase provider to use to obtain the password to use if the proxy server requires authentication. This should be omitted if the proxy server does not require authentication.

For example, you can use a configuration change like the following to create an HTTP proxy external server that does not require authentication:

```
dsconfig create-external-server \
     --server-name "Example HTTP Proxy Server" \
     --type http-proxy \
     --set server-host-name:proxy.example.com \
     --set server-port:3128
```

Configuring server components to use the HTTP proxy external server

Merely defining an HTTP proxy external server in the configuration does not cause the server to use that proxy server for anything. Instead, you must indicate which components should use that proxy server.

This is necessary because you might only need to use an HTTP proxy server for certain components (for example, it might be necessary when accessing external web services, but not for services on the internal private network).

All of the components that support the use of an HTTP proxy server offer an `http-proxy-external-server` configuration property whose value should be the name of the appropriate HTTP proxy external server definition in the configuration. For example, to update the Pwned Passwords password validator to use the HTTP proxy server defined in the "Example HTTP Proxy Server" configuration object, use a configuration change like the following:

```
dsconfig set-password-validator-prop \
     --validator-name "Pwned Passwords" \
     --set "http-proxy-external-server:Example HTTP Proxy Server"
```

resync command">

The `resync` command

The `resync` command provides bulk synchronization that can be used to verify the synchronization setup. The command 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` command also provides options to write debugging output to a log.

> **Note**
>
> The **`resync`** command 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`** command 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 in `logs/tools/resync-errors.log`. If necessary, the logging location can be changed with the `--logFilePath` option.

Test attribute and DN maps

The `resync` command 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` command 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` command 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 PingDirectory servers) 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` command can be run with one pass through the entries.

Use the `resync` command 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` command to populate an empty Sync Destination topology for small datasets. For large deployments, use the `bin/translate-ldif` command.

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 in `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 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 command operates at the maximum rate.

Run the `resync` command first at 100 operations per second, measure the impact on the source servers, then adjust as needed. 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 the `ldapsearch` command 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 `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 PingDirectory Server Administration Guide for details.

Start real-time synchronization globally

About this task

The `realtime-sync` command assumes that the synchronization topology is configured correctly.

Perform the following steps to start real-time synchronization globally:

Steps

1. Run the command 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` option. If the `--pipe-name` option 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 before 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 before starting real-time synchronization. It should be run before 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 before 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 the `ldapsearch` command 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
ltZTo6IEFBQUJKZ25OTUM4PQotCgA=
changenumber: 2329
```

2. Restart synchronization from change number 2329 using the `realtime-sync` command. 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 before 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` command features both an offline mode of operation as well as the ability to schedule an operation to run within the PingDataSync server's process. To schedule an operation, supply LDAP connection options that allow this command 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` command.

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` option.

   ```
   $ bin/realtime-sync set-startpoint \
     --pipe-name "sun-to-ds-sync-pipe" \
     --port 389 \
     --bindDN "uid=admin,dc=example,dc=com" \
     --bindPassword secret \
     --start 20160721000100 \
     --no-prompt
   ```

   ```
   Set StartPoint task 2009072016103807 scheduled to start Jul 21, 2016
   12:01:00 AM CDT
   ```

2. Run the `manage-tasks` command 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 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
```

The Sync Destination could have the standard X.500 DN format.

DN mappings allow the use of wildcards for DN transformations. A single wildcard (`*`) matches a single RDN component and can be used any number of times. The double wildcard (`**`) matches zero or more RDN components and can be used only once.

> **Note**
>
> If a literal '*' is required in a DN, it must be escaped as `'\2A'`.

The wildcard 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 automatically validates any DN mapping before applying the configuration.

Configuring a DN map by using `dsconfig`

About this task

You can configure a DN map by 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.

```
$ 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={1},{2},dc=example,dc=com" \
  --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
```

About attribute mappings

Attribute mappings define how the values of a single destination attribute are determined.

The destination attribute in an attribute mapping can be derived from a direct mapping with a source attribute, constructed from a combination of hard-coded strings and elements of other attributes, or generated using custom Java class code. All attribute mapping types support the following properties:

Property	Description
`to-attribute`	Specifies the name of the attribute whose values are constructed by the mapping. This property is required.
`description`	Describes the attribute mapping. This property is optional.
`exclude-value`	Specifies a list of values to exclude from the destination attribute after applying the mapping. This property is optional.
`also-depends-on-src-attribute`	Specifies the source attributes that trigger an update to the relevant destination attributes when changed, regardless of changes to attributes the mapping directly depends on. This property is optional.
`sync-on-every-update`	Synchronizes the value of `to-attribute` on any detected change of any attribute. This property is optional. **Note** The attribute mapping of `to-attribute` is always evaluated during sync, even if the value didn't change.

Note

Use the **`dsconfig list-attribute-mappings`** command to view configured attribute mappings.

The PingDataSync server supports the following attribute mappings:

JSON attribute mapping

Use for a destination JSON attribute whose JSON field values are constructed by defining JSON attribute mapping field configuration objects. If any field value cannot be constructed, either because of the required source attributes not being present or the resulting value being invalid, that field is omitted from the constructed JSON attribute. The following example creates a JSON attribute mapping with the field `formatted` from the `cn` attribute:

```
$ bin/dsconfig create-attribute-map \
  --map-name PingDirectory_to_PingOne_User_Map
```

```
$ bin/dsconfig create-attribute-mapping \
  --map-name PingDirectory_to_PingOne_User_Map \
  --mapping-name name \
  --type json
```

```
$ bin/dsconfig create-json-attribute-mapping-field \
  --map-name PingDirectory_to_PingOne_User_Map \
  --mapping-name name \
  --field-name formatted \
  --set from-attribute:cn \
  --set json-type:string
```

> **Note**
>
> If any existing JSON fields of the destination attribute must be preserved, a JSON attribute configuration object must be created for the sync class.

Constructed attribute mapping

Use when a destination attribute's values are constructed from static text and multiple source attribute values. The source attribute values can be modified using regular expressions and replacement values. You can use a constructed attribute mapping to provide a fixed set of attribute values or to augment existing attribute values with additional fixed values. Constructed attribute mappings support the following properties:

Property	Description
`value-pattern`	**Specifies a pattern for constructing the destination attribute value using fixed text and attribute values from the source entry.** **Note** You can't have more than one multivalued attribute. If an attribute is multivalued, the destination value takes the same number of values minus duplicates. For example, a mapping of `{givenName}{sn}` produces values of `'Jim Smith'`, `'James Smith'` if `givenName` has values of `'Jim'`, `'James'` and `sn` has a value of `'Smith'`. **This property is optional.**
`conditional-value-pattern`	**Specifies a pattern for conditionally constructing the destination attribute value using fixed text and attribute values from the source entry. The pattern constructs the destination attribute only if the specified LDAP filter matches the source entry. The value of this property has the form `LDAP-filter : pattern`.** **Note** The LDAP filter is similar to what you can pass to the **`ldapsearch`** command. For example, if you want to select the `admin-user-name` attribute from the source only if the `is-admin` attribute has a value of `true`, you can use a value pattern of `(is-admin=true) : {admin-user-name}`. **This property is optional.**

Direct attribute mapping

Use when a destination attribute receives its values directly from a source attribute. To use a direct attribute mapping, the attribute values must not change in format from the source to the destination. Direct attribute mappings support the following properties:

Property	Description
`from-attribute`	**Specifies the name of the source attribute whose values are used to provide the values of the destination attribute.** **This property is required.**

Property	Description
`base64-encode-value`	Encodes the source attribute with base-64 before synchronizing it to the destination. This property is optional.
`base-64-decode-value`	Decodes the source attribute from base-64 before synchronizing it to the destination. This property is optional.

DN attribute mapping

Use when a destination attribute receives its values directly from a source attribute whose distinguished name (DN) values require translation. This translation can be performed either by an existing DN map or a map configured within the attribute mapping itself. DN attribute mappings support the following properties:

Property	Description
`from-attribute`	Specifies the name of the source attribute whose values are used to directly provide the values of the destination attribute. This property is required.
`dn-map`	Constructs the destination DN value using components of the source DN and attributes from the source entry. If source DNs match different DN patterns, you can specify multiple DN maps. This property is optional.

Configure synchronization with JSON attribute values

PingDataSync 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}
```

ubidEmailJSON fully">

Synchronizing `ubidEmailJSON` fully

If a source JSON attribute value should be fully synchronized to a destination JSON attribute value, no special configuration is required.

Synchronizing a subset of fields from the source attribute

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"}
```

You can filter JSON field values in the `value-pattern` and `conditional-value-pattern` constructed attribute mapping properties. This enables a given field value to only be mapped for values that match a given filter.

For example, if `ubidEmailJSON` is a JSON attribute and the `value` field is to be extracted only for values where the `type` field is equal to `work`, the value pattern specifying this would be as follows:

```
{ubidEmailJSON.value({{"filterType":"equals","field":"type","value":"work"}})}
```

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.

Retaining 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.

Synchronizing 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
```

Synchronizing 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, see 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
```

Synchronizing 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 PingDirectory server or a 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

PingDataSync supports fractional replication to any server type. For example, if a replica only performs user authentications, PingDataSync 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 the servers synchronized.

About this task

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 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 Sync Class Properties menu, type the number specifying the source LDAP filter (`include-filter` property) that defines which source entries are to be included in the Sync Class.
5. On the Include-Filter Property menu, type the number corresponding to adding a filter value. For this example, type (`objectclass=person`). When prompted, enter another filter. Press Enter to continue. On the menu, enter 1 to use the value when specifying it.
6. On the Sync Class Properties menu, type the number corresponding to the `auto-mapped-source-attribute` property. Change the value from " `-all-` " to a specific attribute, so that only the specified attribute is automatically mapped from the source topology to the destination topology.
7. On the Auto-Mapped-Source-Attribute Property menu, type the number corresponding to adding the source attributes that will be automatically mapped to the destination attributes of the same name. When prompted, enter each attribute, and then press Enter.

   ```
   Enter another value for the 'auto-mapped-source-attribute' property
   [continue]: uid
   Enter another value for the 'auto-mapped-source-attribute' property
   [continue]: userPassword
   Enter another value for the 'auto-mapped-source-attribute' property
   [continue]: objectclass
   Enter another value for the 'auto-mapped-source-attribute' property
   [continue]:
   ```

8. On the Auto-Mapped-Source-Attribute Property menu, type the number corresponding to removing one or more values. In this example, remove the " `-all-` " value, so that only the `objectclass` , `uid` , and `userPassword` attributes are 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 might 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.

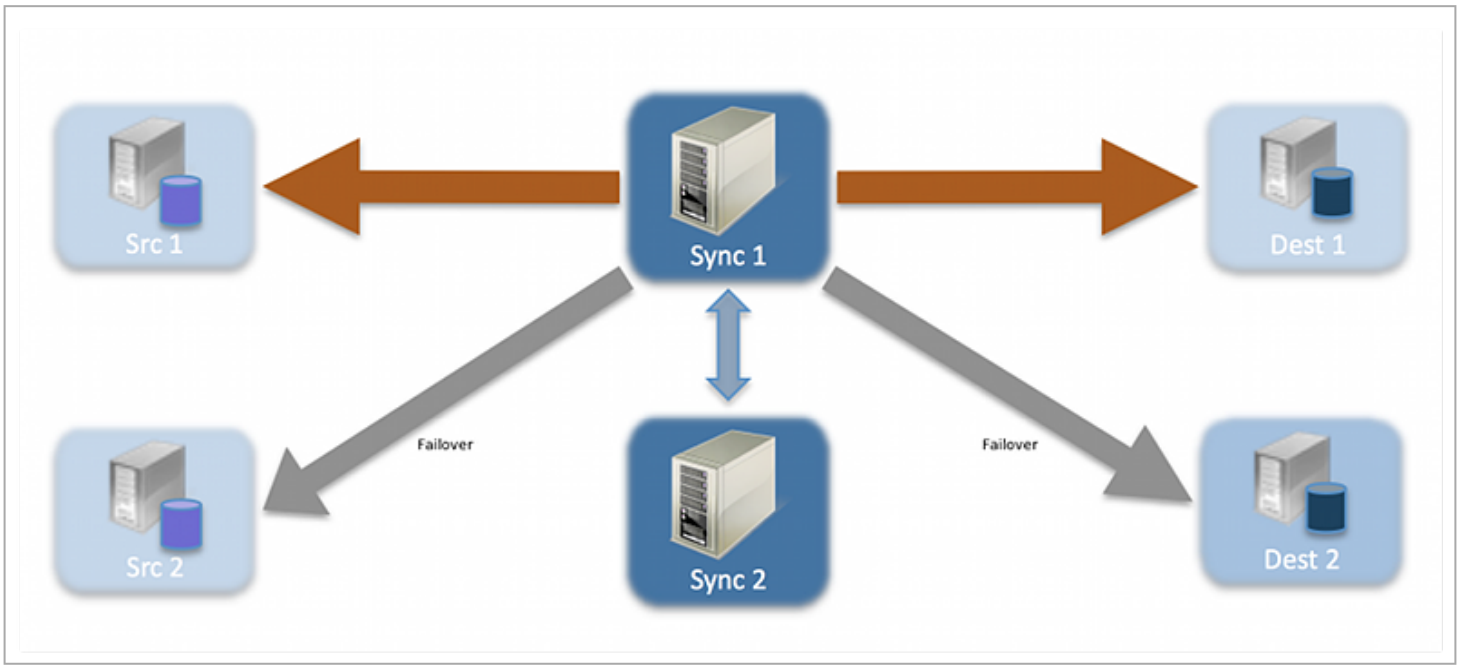
```
$ bin/realtime-sync start --pipe-name "test-sync-pipe" \
  --port 7389 \
  --bindDN "uid=admin,dc=example,dc=com" \
  --bindPassword secret \
  --no-prompt
```

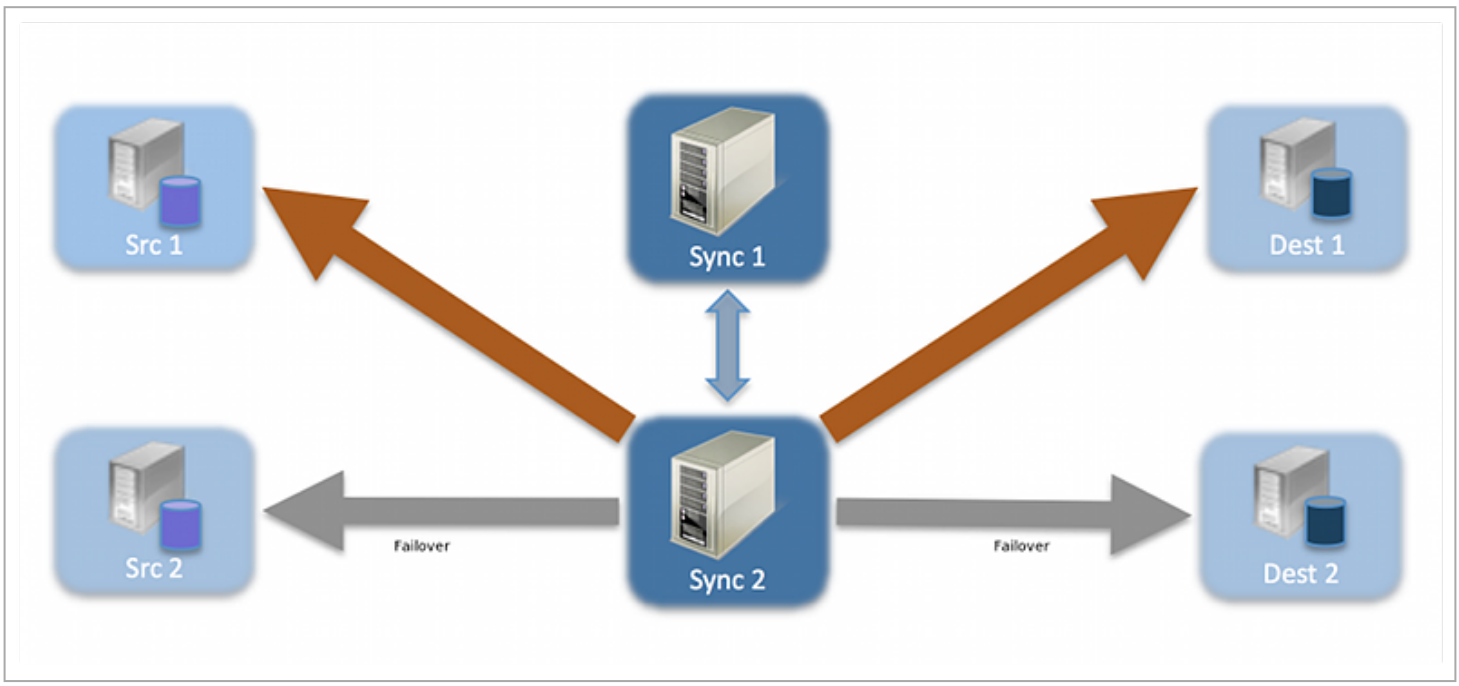
Configure failover behavior

The following figure 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 after 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 until the sync pipe is restarted, regardless of whether the main external server comes back online.





Triggering failover in a topology

For PingDataSync servers in a topology, you can trigger a failover to another server instance if the active server isn't operating as expected.

Steps

- Trigger a failover to another PingDataSync server instance:

  Choose from:

  - Stop the active server instance by running the `stop-server` command.
  - Increase the value of the `sync-server-priority-index` property to lower the priority of the active server instance.

    Example:

    ```
    $ bin/dsconfig set-external-server-prop \
        --server-name intra-sync-<server:port> \
        --set sync-server-priority-index:3 \
        --no-prompt --applyChangeTo server-group
    ```

    > **Note**
    >
    > Learn more about the `sync-server-priority-index` property in the PingDataSync configuration documentation.

Conditions that trigger immediate failover

Immediate failover occurs when PingDataSync receives one of the following error codes from an external server:

- BUSY (51)
- UNAVAILABLE (52)
- SERVER CONNECTION CLOSED (81)
- CONNECT ERROR (91)

If PingDataSync attempts a write operation to a target server that returns one of these error codes, PingDataSync will automatically fail over to the server instance with the next-highest priority 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 supports endpoint failover, which is configurable using the `location` property on the external servers. By default, PingDataSync prefers to connect to and failover to endpoint servers in the same location as itself. If no location settings are configured, PingDataSync will iterate through the configured list of external servers on the Sync Source and Sync Destination when failing over.

PingDataSync does not perform periodic health checks of external servers, and does not failover automatically to a preferred external server. Because of the cost of sync failover, PingDataSync remains connected to a given server until the server stops responding or until the Sync Pipe is restarted. When a failover occurs, PingDataSync returns to the most preferred server, optionally using location settings to identify it, and works its way down the list. Here is 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 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 iterates through the list, it returns to the top of the list. If PingDataSync 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. Assume that PingDataSync 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 gets its changes from server austin1. If austin1 goes down, PingDataSync 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 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. If boston1 is unavailable, PingDataSync 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.

Error Code	Description
CLIENT_SIDE_LOCAL_ERROR(82)	Indicates that a local (client-side) error occurred.
CLIENT_SIDE_ENCODING_ERROR(83)	Indicates that an error occurred while encoding a request.
CLIENT_SIDE_DECODING_ERROR(84)	Indicates that an error occurred while decoding a request.
CLIENT_SIDE_TIMEOUT(85)	Indicates that a client-side timeout occurred.
CLIENT_SIDE_USER_CANCELLED(88)	Indicates that a user canceled a client-side operation.
CLIENT_SIDE_NO_MEMORY(90)	Indicates that the client could not obtain enough memory to perform the requested operation.
CLIENT_SIDE_CLIENT_LOOP(96)	Indicates that a referral loop was 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.

max-operation-attempts property">

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
```

response-timeout property">

The `response-timeout` property

The `response-timeout` property specifies how long PingDataSync 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 "response-timeout:9 s"
```

max-failover-error-code-frequency property">

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"
```

max-backtrack-replication-latency property">

The `max-backtrack-replication-latency` property

The `max-backtrack-replication-latency` property (part of the Sync Source configuration) sets the time period that PingDataSync will look for missed changes in the change log caused by 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 stops looking in the change log after it finds a change that is older than the maximum replication latency.

For example, after failing over to another server, PingDataSync must look through the new server's change log to find the equivalent place to begin synchronizing changes.

Normally, PingDataSync 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 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` command.
- 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 PingDataSync 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 PingDataSync server. See the product documentation for the device type. For PingDataSync 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 authenticates to the PingDirectory server using LDAP simple authentication (with a bind DN and a password). However, PingDataSync can be configured to use SASL EXTERNAL to authenticate to the PingDirectory server with a client certificate.

> **Note**
>
> This procedure assumes that PingDataSync 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 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 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 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 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 should re-establish connections to the LDAP external server and authenticate with SASL EXTERNAL. Verify that PingDataSync 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 to make sure that the BIND RESULT log messages used to authenticate the connections from PingDataSync 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 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 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 will always fail over to servers that are in the same location. If there are multiple Sync Sources in the same location as PingDataSync, then PingDataSync 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, 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`** command 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`** command 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 PingDirectory 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

PingDataSync supports PingOne as a synchronization destination and source for newly created or modified accounts with native password changes between directory servers, relational databases, or other PingOne systems.

> **Note**
>
> For PingOne documentation, see PingOne Platform.

The following topics include configuration procedures for synchronization between PingOne and the PingDirectory server, or other LDAP source servers or targets.

- Prerequisites
- Synchronize changes to a PingOne environment
- Synchronize changes from a PingOne environment
- Testing and troubleshooting the sync
- PingOne synchronization limitations

For more information, see Syncing passwords to PingOne.

Prerequisites

Before attempting to synchronize changes to or from a PingOne environment, make certain the prerequisites in this section are satisfied.

Create a worker application

A worker application is an administrator application that can have the same roles as human administrators. Creating a worker application creates a sync destination or source for synchronizing changes to and from PingOne.

Before you begin

Before you create a worker application, have the following information ready:

- The app name and description
- Redirect URLs for authentication (required for interactive applications only)

About this task

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.

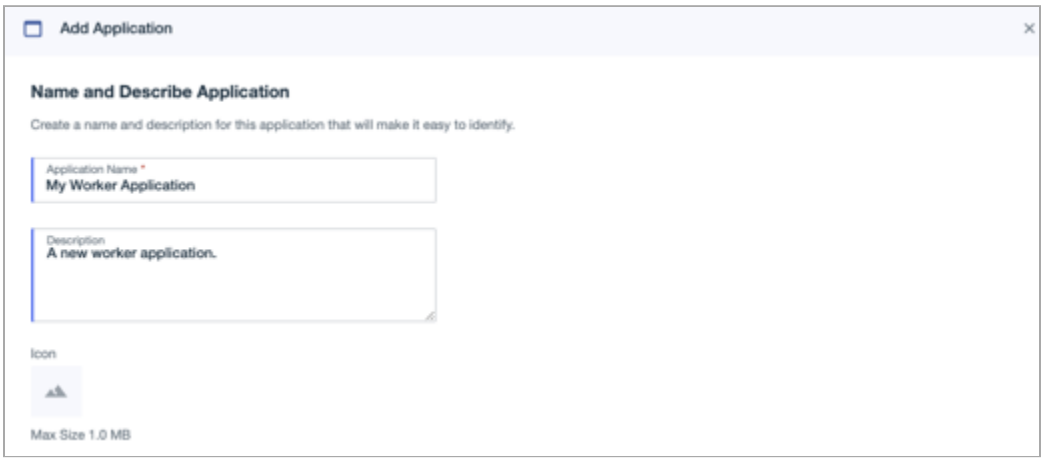
Required roles

A role is a collection of permissions that can be assigned to a user. Of the many roles that PingOne 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.

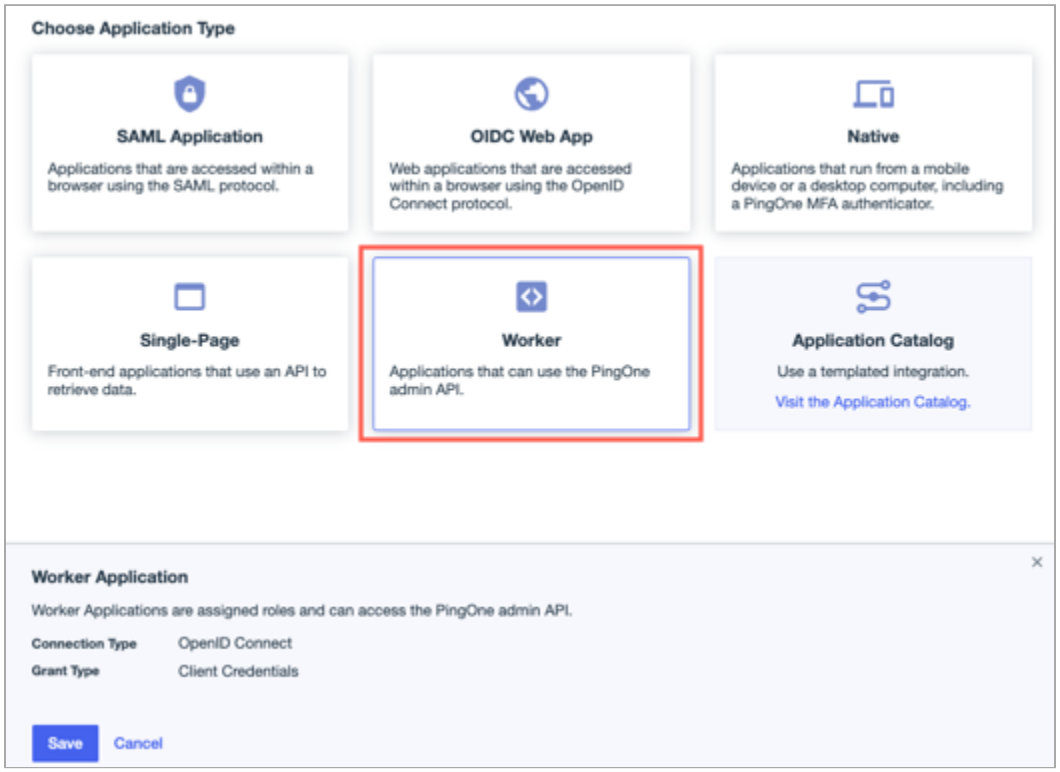
To create and configure a worker app in PingOne:

Steps

1. In the PingOne admin portal, go to Connections → Applications.
2. Click the + icon.
3. Create the application profile by entering the following:
    - Application name: A unique identifier for the application.
    - Description (optional): A brief characterization of the application.
    - Icon (optional): A graphic representation of the application. Use a file up to 1MB in JPG, JPEG, GIF, or PNG format.



4. In the Choose Application Type section, click Worker.



5. Click Save.

   Result:

   The app is displayed on the Applications page.

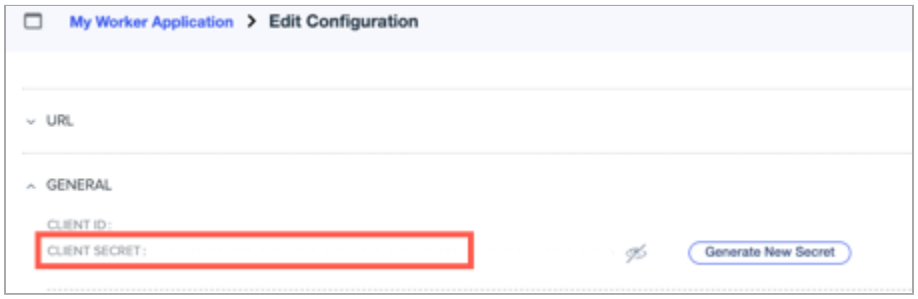
6. Make note of the OAuth Client ID, which appears directly below the name of the app.

   This value is required when creating a PingOne sync destination or source.

7. Click the Configuration tab, and then click the Pencil icon to edit the configuration:

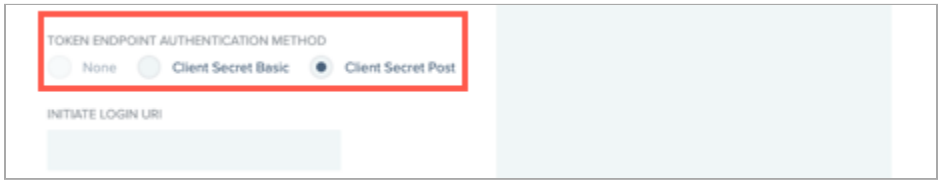
   1. In the General section, make note of the Client Secret.

      This value is required when creating a PingOne sync destination or source.



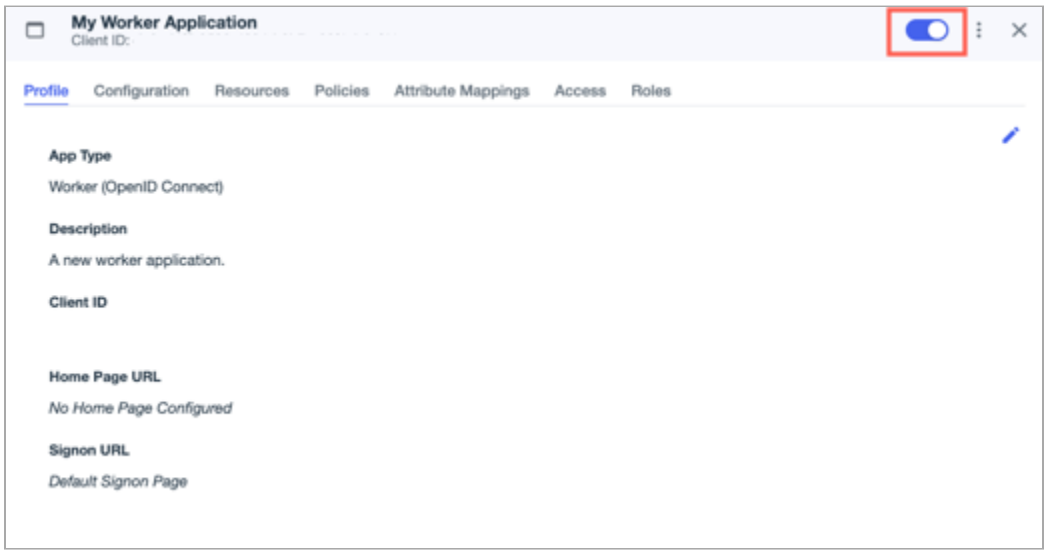
   2. For Grant Type, select the Client Credentials check box.

3. For a token endpoint authentication method, click Client Secret Post.



4. Click Save.

8. Click the toggle to enable the application.



9. In the left navigation pane, click Environment → Properties.

10. Make note of the Environment ID.

    This value is required when creating a PingOne sync destination or source.

Next steps

Use your worker application as a sync source or destination for PingOne:

- Synchronize changes to a PingOne environment.
- Synchronize changes from a PingOne environment.

Review the 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, see the PingOne Platform API Reference.

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.

Synchronize changes to a PingOne environment

This section describes the configuration that is necessary to synchronize changes to a PingOne environment. PingDataSync supports synchronization of single and multivalued attributes to PingOne. To view an example configuration, see the file located at `<server-root>/config/sample-dsconfig-batch-files/reference-ping-one-sync-destination-configuration.dsconfig`.

> **Note**
>
> When configuring a sync pipe in PingDataSync to synchronize users to a PingOne destination, you must include a constructed attribute mapping named `resourceType` with a value-pattern of `user`.

Creating a PingOne sync destination

Before you create a PingOne sync destination, make sure you have the following information ready:

- Environment ID (*environment-id*)
- OAuth client ID (*oauth-client-id*)
- OAuth client secret (*oauth-client-secret*)

Learn more about obtaining these values in Creating a PingOne worker application.

The following sample creates a PingOne 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]
```

Configuring JSON attribute mappings

Add the JSON attribute mapping type with sub-objects (the JSON attribute mapping field) that allow you to map individual fields.

About this task

If a source attribute doesn't have a value, the corresponding field is omitted.

> **Note**
>
> Use JSON attribute mappings rather than constructed attribute mappings.

Steps

1. To create an attribute map, run `dsconfig` with the `create-attribute-map` option.

   Example:

   The following example creates an attribute map titled `PingDirectory_to_PingOne_User_Map`.

   ```
   dsconfig create-attribute-map \
     --map-name PingDirectory_to_PingOne_User_Map
   ```

2. To create the attribute mapping, run `dsconfig` with the `create-attribute-mapping` option.

   Example:

   The following example creates the attribute mapping to `PingDirectory_to_PingOne_User_Map`.

   ```
   dsconfig create-attribute-mapping \
     --map-name PingDirectory_to_PingOne_User_Map \
     --mapping-name name \
     --type json
   ```

3. To map JSON attributes, run `dsconfig` with the `create-json-attribute-mapping-field` option.

   Example:

   The following example creates the JSON attribute mapping field `formatted` from the `cn` attribute.

   ```
   dsconfig create-json-attribute-mapping-field \
     --map-name PingDirectory_to_PingOne_User_Map \
     --mapping-name name \
     --field-name formatted \
     --set from-attribute:cn \
     --set json-type:string
   ```

   The following example creates the JSON attribute mapping field `given` from the `givenName` attribute.

```
dsconfig create-json-attribute-mapping-field \
  --map-name PingDirectory_to_PingOne_User_Map \
  --mapping-name name \
  --field-name given \
  --set from-attribute:givenName \
  --set json-type:string
```

The following example creates the JSON attribute mapping field `family` from the `sn` attribute.

```
dsconfig create-json-attribute-mapping-field \
  --map-name PingDirectory_to_PingOne_User_Map \
  --mapping-name name \
  --field-name family \
  --set from-attribute:sn \
  --set json-type:string
```

Configuring constructed attribute mappings

> **Note**
>
> You should use JSON attribute mappings instead of constructed attribute mappings. Learn more in Configuring JSON attribute mappings.

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, see the PingOne Platform API Reference.

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

There are limitations and other constraints to consider when synchronizing changes to a PingOne environment.

Renaming entries isn't allowed

As a sync destination, PingOne doesn't allow `moddn` operations to rename an entry. When creating a sync class, set the `allow-destination-renames` option to `false` to prevent PingDataSync from attempting to rename entries and then logging related errors. For example:

```
dsconfig create-sync-class \
  --pipe-name PingDirectory_to_PingOne \
  --class-name "PingDirectory_to_PingOne User Sync Class" \
  --set allow-destination-renames:false \
  --set attribute-map:PingDirectory_to_PingOne_User_Map \
  --set auto-mapped-source-attribute:-none- \
  --set destination-correlation-attributes:externalId \
  --set destination-correlation-attributes:username \
  --set destination-correlation-attributes:email \
  --set destination-create-only-attribute:resourceType \
  --set creates-as-modifies:true \
  --set include-filter:(objectClass=person)
```

Tip

You can find the previous example in the PingDataSync distribution. Learn more in the following file: `config/sample-dsconfig-batch-files/reference-pingone-sync-destination-configuration.dsconfig` .

Populations

Note

All PingOne user resources must exist within a population.

The PingOne 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 with a constructed attribute mapping:

```
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, construct a valid JSON object.

Multivalued attributes

If your incoming data is in JSON format, configure your PingOne multivalued attribute as JSON and use a JSON attribute mapping.

If your incoming data is not in JSON format, you can configure your PingOne multivalued attribute as JSON and use a constructed attribute mapping. Otherwise, you must configure your PingOne multivalued attribute as `DECLARED` and use a direct attribute mapping.

> **Important**
>
> Direct attribute mapping does not work with JSON multivalued PingOne attributes even with an attribute with the same name and value in PingDirectory.

Synchronize changes from a PingOne environment

This section describes the configuration that is necessary to synchronize changes from a PingOne environment. To view an example configuration, see the file located at `<server-root>/config/sample-dsconfig-batch-files/reference-ping-one-sync-source-configuration.dsconfig`.

Create a PingOne sync source

Before you create a PingOne 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*)

Learn more about obtaining these values in Creating a PingOne worker application.

The following example creates a PingOne 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]
```

Configuring attribute mapping

The process of synchronizing data uses 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 an attribute mapping. 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.

Synchronizing PingOne account status with PingDirectory

About this task

When configuring a one-way sync from PingOne to PingDirectory, you can enable PingDirectory to synchronize with the `User.Locked` or `User.Unlocked` status of a PingOne account. This synchronization reduces confusion for the administrator and enhances the security posture of the account. For example, if a PingOne account is locked due to multiple failed MFA attempts, it could be risky to leave the corresponding account fully functional.

> **Note**
>
> A PingOne `User.Locked` event disables the destination entry in PingDirectory. A `User.Unlocked` event enables the destination entry.

To enable this synchronization, you must map two account status attributes from PingOne directly to the corresponding PingDirectory attributes. Because the PingDirectory attributes can't be written to directly, PingDataSync uses intermediate attributes to facilitate an extended operation.

The following table shows the relevant source, intermediate, and destination attributes for this mapping:

PingOne attribute	Intermediate attribute	PingDirectory attribute
`account.status`	`ds-pwp-account-disabled-from-pingone`	`ds-pwp-account-disabled`
`account.lockedAt`	`pwdAccountLockedTimeFromPingOne`	`pwdAccountLockedTime`

> **Note**
>
> Intermediate attributes only exist in memory on the PingDataSync server so that they can be consumed for attribute mappings. They don't exist in PingOne or on the PingDirectory server.

Steps

- Map `ds-pwp-account-disabled-from-pingone` to `ds-pwp-account-disabled` using the following command:

```
dsconfig create-attribute-mapping \
  --map-name PingOne_to_PingDirectory_User_Map \
  --mapping-name ds-pwp-account-disabled \
  --type direct \
  --set from-attribute:ds-pwp-account-disabled-from-pingone
```

- Map `pwdAccountLockedTimeFromPingOne` to `pwdAccountLockedTime` using the following command:

```
dsconfig create-attribute-mapping \
  --map-name PingOne_to_PingDirectory_User_Map \
  --mapping-name pwdAccountLockedTime \
  --type direct \
  --set from-attribute:pwdAccountLockedTimeFromPingOne
```

- To have the `pwdAccountLockedTime` attribute appear on the destination entry, set the `lockout-failure-count` attribute to a non-zero value.

  Example:

  ```
  dsconfig set-password-policy-prop \
    --policy-name "Default Password Policy" \
    --set lockout-failure-count:1
  ```

Troubleshooting

By default, the `modifies-as-creates` sync class property is set to `false`.

Active Directory attributes might not be synchronized as expected when all of the following are true:

- You are using the `realtime-sync` tool.
- The `modifies-as-creates` sync class property is set to `true`.
- A modification is detected on the source endpoint to a missing entry on the destination endpoint.
- The modification is to attributes other than the two PingOne attributes previously mentioned.

To avoid this, you can run the `resync` tool instead of the `realtime-sync` tool. Using `resync` will correctly copy all attributes. Learn more about the resync command.

Considerations and limitations

This section describes limitations and other constraints to consider when synchronizing changes from a PingOne environment.

Bidirectional synchronization

If you plan on configuring bidirectional synchronization between PingOne and PingDirectory, make sure 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 does not support the synchronizing of passwords from PingOne.

Population management

If your PingOne 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 propagates changes throughout PingOne 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 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.

Testing and troubleshooting the sync

Follow these steps to test and troubleshoot your configured sync between PingOne and PingDirectory.

Steps

1. Test the sync:
    1. Run the sync with the following command:

        ```
        /opt/<PingDataSync>/bin/resync -p  <sync_pipe_name>
        ```

    2. (Optional) If the sync results in any errors, see the `/Ping/<PingDataSync>/logs/tools/re-sync-failed-DNs.log`.
    3. (Optional) If you receive an error that includes `Cannot connect because: The connection to server localhost:11389 was closed while waiting for a response to a bind request SimpleBindRequest(dn='cn=dmanager').`:
        1. In the PingDataSync admin console, go to Configuration → External Servers → ServerPD_PDtest.
        2. Update your password.

PingOne synchronization limitations

PingOne can return errors when synchronizing with PingDataSync because of a rate-limiting mechanism in PingOne. A workaround for this issue has been implemented in PingDataSync.

The PingOne API contains a resource-limit mechanism to prevent clients from overwhelming the service and returns HTTP 429 error codes when this limit is reached.

> **Note**
>
> This limit is set to 100 operations per second.

Because PingDataSync sends changes to a destination as quickly as possible, the server might hit this limit, especially if a large number of threads are in use.

If the PingDataSync server receives HTTP 429 errors from PingOne, it waits briefly and retries any failed operations again. An operation is retried up to 15 times to attempt to synchronize a change. This retry mechanism allows the PingDataSync server to process a change without the operation dropping as a failure after initial HTTP 429 errors.

Synchronize with Active Directory and other directory servers

PingDataSync supports full synchronization for newly created or modified accounts with native password changes between directory server, relational databases, and Microsoft Active Directory (AD) systems.

Considerations

There are three key considerations when synchronizing between AD and PingDirectory:

The `realtime-sync` tool

The `realtime-sync` tool uses the AD DirSync control to detect changes on entries, which requires the control to be searched at the top of the directory information tree (DIT). Because of this, you must point your AD Sync Source to the top of the AD tree for `realtime-sync` to work.

Distinguished name (DN) mapping

The AD Sync Source must be pointed at the top of the DIT, but not every branch under the top of the tree can be easily synchronized.

For example, `cn=Users` is a container organizational unit (OU) that doesn't easily convert into a standard OU. Likewise, `cn=Builtin` is a top-level domain that also contains built-in groups without a purpose in PingDirectory and that don't need to be synchronized.

To avoid synchronizing entries that are native and apply only to AD, point your Sync Classes at specific OUs.

Schema and attribute mapping

The schema between AD and PingDirectory is not a 1:1 relationship, which means that not all attributes can be directly synchronized.

The following attributes are among those that can be directly synchronized:

- `cn`
- `sn`
- `mail`

Other attributes, such as the AD attribute `{{samAccountName}}` aren't defined in PingDirectory by default, and if you don't define schema for the attribute, you can map it to a similar attribute such as the PingDirectory `uid` attribute. You should create attribute mappings for each attribute that you want to synchronize between AD and PingDirectory.

Known limitations and workarounds

Tracking group membership changes in AD

The virtual attribute `memberOf` exists in an AD user entry and contains a list of that user's group memberships. When group membership changes, AD updates only the group entry member attribute. Therefore, if PingDataSync monitors only `memberOf` for group membership changes in AD, it won't detect them.

You can try the following workarounds:

- Run the `resync` command periodically.
- Manually sync the groups between AD and PingDirectory.

> **Important**
>
> This requires the ability to map DNs between AD and PingDirectory based on the available information, which is often limited.

Syncing passwords from LDAP servers to AD

You can sync passwords from PingDirectory to AD, but syncing passwords directly from other Lightweight Directory Access Protocol (LDAP) servers to AD isn't supported. You should sync these passwords to PingDirectory first, which allows you to then sync them to AD.

Configuration information

For configuration information and procedures for synchronization between PingDirectory server or other LDAP source servers or targets with Microsoft AD systems, refer to the following:

- Overview of configuration tasks

From PingDirectory to AD

- Active Directory sync user account
- Preparing external servers
- Configuring password encryption
- Password Sync Agent

From AD to PingDirectory

- Configuring one way synchronization from Active Directory to PingDirectory
- Mapping AD password policy state attributes to PingDirectory using `dsconfig`
- Configuring sync pipes and sync classes

Overview of configuration tasks

PingDataSync supports bidirectional synchronization between PingDirectory and Active Directory (AD). This section describes the configuration tasks that are necessary to synchronize changes to Active Directory systems. To view an example configuration, see the file located in the `<server-root>/config/sample-dsconfig-batch-files/reference-bidirectional-sync-activedirectory-pingdirectory.dsconfig` directory.

Enable SSL connections

If you are synchronizing passwords between systems, Active Directory systems require that SSL be enabled on the Active Directory domain controller, so that PingDataSync can securely propagate the `cn=Sync User` account password and other user passwords to the target.

Run the create-sync-pipe-config tool

On the PingDataSync server, use the `create-sync-pipe-config` tool to configure the Sync Pipes to communicate with the Active Directory source or target.

Configure outbound password synchronization on an PingDirectory Server Sync Source

After running the `create-sync-pipe-config` tool, determine if outbound password synchronization from a 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 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.

Run the realtime-sync set-startpoint tool

The `realtime-sync set-startpoint` command can 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 received a cookie that is up-to-date. NOTE: If the Password Sync Agent is down for any length of time and misses a password change, these changes will not be synced on recovery without either a new password change for the entry or the use of pass-through authentication. The Password Sync Agent cannot be pointed at multiple domain clusters.

Configuring one way synchronization from Active Directory to PingDirectory

Configure a one-way Sync Pipe with the Active Directory (AD) topology as the sync source and a PingDirectory server topology as the Sync Destination.

About this task

Syncing from AD-LDS to PingDirectory is supported for all features except password syncing.

Important

If you are syncing the `lockoutTime`, `userAccountControl & (ACCOUNTDISABLE == 2)`, or `pwdLastSet` AD attributes, or the AD-LDS `ms-DS-User-Account-Disabled` attribute, see Synchronizing Active Directory with PingDirectory.

Note

The Password Sync Agent cannot be pointed at multiple domain clusters.

Steps

1. From the `server-root` directory, start PingDataSync.

```
$ <server-root>/bin/start-server
```

2. To set up the initial synchronization topology, run the `sync` tool.

```
$ bin/create-sync-pipe-config
```

3. In the Create Initial Synchronization Configuration menu, press Enter to continue the configuration.
4. In the Synchronization Mode menu, press Enter to accept the default option `1` for `Standard mode`.
5. In the Synchronization Direction menu, press Enter to accept the default option `1` for `One way`.
6. In the Source Endpoint Type menu, enter option `7` for `Microsoft Active Directory`.
7. In the Source Endpoint Name menu, enter a name for the Microsoft AD source server, or press Enter to accept the default value of `Microsoft Active Directory Source`.
8. In the <*Source Server*> Server Security menu, press Enter to accept the default option `1` for `SSL` security.
9. In the <*Source Server*> Servers menu, enter the host name and listener port for Lightweight Directory Access Protocol (LDAP) communication with the source server in the format of `<host name>:<port number>` and press Enter.

   The Data Sync server attempts a connection to the AD source server. After adding the first server, you can add additional servers for the source endpoints that will be prioritized below the first server.
10. When you have finished adding servers, press Enter to continue to the next configuration step.
11. In the Synchronization User Account for <*Source Server*> menu, enter a user account distinguished name (DN) for the source servers, or press Enter to accept the default value.

    The account is used exclusively by the Data Sync Server to communicate with the source external servers.
12. Enter a password for the synchronization user account and press Enter.

    Note

    The User Account DN password must meet the minimum password requirements for AD domains.

13. In the Destination Endpoint Type menu, press Enter to select the default option `1` for `Ping Identity Directory Server`.
14. In the Destination Endpoint Name menu, enter a name for your destination endpoint, or press Enter to select the default value, `Ping Identity Directory Server Destination`.
15. In the Base DNs for *<Endpoint Server>* menu, enter a base DN where synchronized entries can be found in your endpoint server, or press Enter to accept the default value.

    After your initial entry, you can add additional base DNs by following the prompts.
16. When you have finished entering base DNs for synchronized entries, press Enter to continue the configuration.
17. In the *<Endpoint Server>* Server Security menu, enter the option for the type of security that the Sync Server will use in communication with the endpoint server and press Enter.
18. In the *<Endpoint Server>* Servers menu, enter the host name and port for LDAP communication in the format of `<host name>:<port number>` and press Enter.

    The PingDataSync server attempts a connection to the destination PingDirectory server endpoint. After adding the first server, you can add additional servers for the destination endpoints that will be prioritized below the first server.
19. When you have finished adding servers, press Enter to continue to the next configuration step.
20. In the Synchronization User Account for *<Endpoint Server>* menu, enter a DN for the synchronization user account that will be used in communication with external servers, or press Enter to accept the default value, `[cn=Sync User,cn=Root DNs,cn=config]`.
21. Enter a password for the synchronization user account and press Enter.
22. In the Prepare Server *<Source Server>* menu, press Enter to accept the default option `1` for `Yes` to prepare the source server for synchronization.
23. In the Prepare Server *<Endpoint Server>* menu, press Enter to accept the default option `1` for `Yes` to prepare the endpoint server for synchronization.
24. In the Sync Pipe Name menu, enter a name for the Sync Pipe from the source server (AD) to the endpoint server (PingDirectory server), or press Enter to select the default value, `Microsoft_Active_Directory_Source_to_Ping_Identity_Directory_Server_Destination`.
25. In the Pre-configured Sync Class Configuration for Active Directory Sync Source menu, follow the prompts to create the basic sync classes and attribute mappings needed to synchronize user accounts, user passwords, and groups to and from AD.

    1. To synchronize user **`Create`**, **`Modify`**, and **`Delete`** operations from AD, follow the prompts.
    2. Enter the object class for user entries at the endpoint, or press Enter to accept the default value, `inetOrgPerson`.
    3. To configure which password policy state attributes to synchronize, follow the prompts.

       For more information on the AD to PingDirectory password policy state attribute mappings, see Synchronizing Active Directory with PingDirectory.

> **Note**
>
> For the referenced password policy state attributes, AD is treated as the authoritative source, because synchronization from PingDirectory to AD is not supported for those attributes.

> **Important**
>
> The password policy in PingDirectory must match the password in AD. For example, the `lockout-failure-count` in PingDirectory must match the account lockout threshold in AD.

4. To create a DN map for users in the sync pipe, enter `yes` and press Enter. To not create a DN map, press Enter to accept the default option, `no`.
5. Review the list of basic mappings set up for synchronized user entries and follow the prompts to add any additional attribute mappings. Press Enter to continue.
6. To synchronize group `Create`, `Modify`, and `Delete` operations from AD, follow the prompts.

26. In the Sync Pipe Sync Class Definitions menu, either press Enter to accept the `Microsoft Active Directory Source Users Sync Class`, or enter a value and press Enter to create a new sync class name.
27. Review the Configuration Summary and press Enter to write the configuration file as displayed.

    Result:

    The server writes the configuration file to a `dsconfig` batch file.
28. To apply the configuration changes to the local PingDataSync server, press Enter. (If you don't want to apply the changes, enter `no` and press Enter.)

Synchronizing Active Directory with PingDirectory

When you use the `sync-pipe` tool to configure AD or AD-LDS as a one-way sync with PingDirectory, three AD password policy state attributes require user input to map to a corresponding PingDirectory attribute.

The following table shows these three attributes, the intermediate attribute that is formed between PingDirectory and AD (or AD-LDS), and the extended operation type used by the PingDirectory server to apply the change.

AD and AD-LDS attribute	Intermediate attribute	PingDirectory attribute	PasswordPolicyStateOperation opType
`lockoutTime`	`pwdAccountLockedTimeFromAD`	`pwdAccountLockedTime`	`OP_TYPE_SET_AUTH_FAILURE_TIMES`
`userAccountControl & (ACCOUNTDISABLE == 2)` **Note** In AD-LDS, the corresponding attribute is `msDS-User-Account-Disabled`.	`ds-pwp-account-disabled-from-ad`	`ds-pwp-account-disabled`	`OP_TYPE_SET_ACCOUNT_DISABLED_STATE`

AD and AD-LDS attribute	Intermediate attribute	PingDirectory attribute	PasswordPolicyStateOperation opType
`pwdLastSet`	`pwdChangedTimeFromAD`	`pwdChangedTime`	`OP_TYPE_SET_PW_CHANGED_TIME`

> **Note**
>
> Intermediate attributes only exist in memory on the PingDataSync server so that they can be consumed for attribute mappings. They don't exist on either the AD server or on the PingDirectory server.

`modifies-as-creates`

By default, the `modifies-as-creates` sync class property is set to `false`.

Active Directory attributes might not be synchronized as expected when the following is true:

- **You are using the `realtime-sync` tool.**
- **The `modifies-as-creates` sync class property is set to `true`.**
- **A modification is detected on the source endpoint to a missing entry on the destination endpoint.**
- **The modification is to attributes other than the three AD password policy state attributes previously mentioned.**

To avoid this known issue, you can run the `resync` tool instead of the `realtime-sync` tool. Using `resync` will correctly copy all attributes. For more information, see The `resync` command.

dsconfig">

Mapping AD password policy state attributes to PingDirectory using `dsconfig`

If you have a working sync configuration between PingDirectory and Active Directory (AD) and want to manage password policy state attributes, use the `dsconfig` command to map these attributes instead of re-running the `sync` command.

About this task

To map AD password policy state attributes to PingDirectory attributes:

Steps

- **Run `dsconfig` with the `create-attribute-mapping` option.**

  Example:

  The following example maps the AD attribute `lockoutTime` to the PingDirectory attribute `pwdAccountLockedTime`.

```
dsconfig create-attribute-mapping
        --map-name  "<Microsoft Active Directory Users Attribute Map>"
        --mapping-name pwdAccountLockedTime
        --type direct
        --set from-attribute:pwdAccountLockedTimeFromAD
```

Example:

The following example maps the AD attribute `userAccountControl & (ACCOUNTDISABLE == 2)` to the PingDirectory attribute `ds-pwp-account-disabled`.

```
dsconfig create-attribute-mapping
        --map-name  "<Microsoft Active Directory Users Attribute Map>"
        --mapping-name ds-pwp-account-disabled
        --type direct
        --set from-attribute:ds-pwp-account-disabled-from-ad
```

Example:

The following example maps the AD attribute `pwdLastSet` to the PingDirectory attribute `pwdChangedTime`.

```
dsconfig create-attribute-mapping
        --map-name  "<Microsoft Active Directory Users Attribute Map>"
        --mapping-name pwdChangedTime
        --type direct
        --set from-attribute:pwdChangedTimeFromAD
```

> **Note**
>
> Learn more about synchronizing these AD attributes with PingDirectory in Synchronizing Active Directory with PingDirectory.

Active Directory sync user account

The Sync User created for Active Directory (AD) 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 PingDataSync server to see deleted entries, the user account must have sufficient access to `cn=Deleted Objects,<domain name>`. Giving access to that distinguished name (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: NOTE: The Password Sync Agent cannot be pointed at multiple domain clusters.

Steps

1. After configuring the source and destination endpoints, PingDataSync 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` command.
2. Configuring this server for synchronization requires manager access. Enter the distinguished name (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 (AD), 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 distinguished name (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 Destination.

   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` command. 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 ->  {pingdir}
   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
   ```

5. 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
   ```

6. If synchronizing group CREATE, MODIFY, and DELETE operations from Active Directory, enter `yes`.

7. Review the basic user group mappings.

8. On the Sync Pipe Sync Class Definitions menu, enter another name for a new Sync Class if required. Repeat steps 2–6 to define this new Sync Class. If no additional Sync Class definitions are required, press Enter to continue.

9. 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 their associated Sync Classes.

```
>>>> Configuration Summary
  Sync Pipe: AD to  {pingdir}  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:  {pingdir}  Server
      Type:  {pingdir}  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:  {pingdir}  Server to AD
    Source:  {pingdir}  Server
      Type:  {pingdir}  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:
        {pingdir}  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
```

10. To apply the configuration to the local PingDataSync server instance, type `yes`. The configuration is recorded at `<server-root>/logs/tools/createsync-pipe-config.log`.

Configuring password encryption

You must follow this procedure when synchronizing passwords from a PingDirectory server to Active Directory (AD), or when synchronizing clear text passwords.

About this task

These steps aren't required for the following scenarios:

- Synchronizing from AD to a PingDirectory server
- Excluding password synchronization

Steps

1. On the PingDirectory server that will receive the password modifications, enable the Change Log Password Encryption component. The component intercepts password modifications, encrypts the password and adds an encrypted attribute, `ds-changelog- encrypted-password`, to the change log entry. The encryption key can be copied from the output if displayed, or accessed from the `<serverroot>/bin/sync-pipe-cfg.txt` file.

```
$ bin/dsconfig set-plugin-prop --plugin-name "Changelog Password
Encryption" \
  --set enabled:true \
  --set changelog-password-encryption-key:<key>
```

2. On PingDataSync, 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 the bulk comparison capability of the resync tool. Then, use the `realtime-sync` tool to start synchronizing the data. If synchronizing passwords, install the Password Sync Agent (PSA) on all of the domain controllers in the topology.

> **Note**
>
> To synchronize passwords from PingDirectory to AD, you must use the **`realtime-sync`** tool. The **`resync`** tool isn't supported for this operation.

Password Sync Agent

When synchronizing passwords with Active Directory (AD) systems, PingDataSync 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.

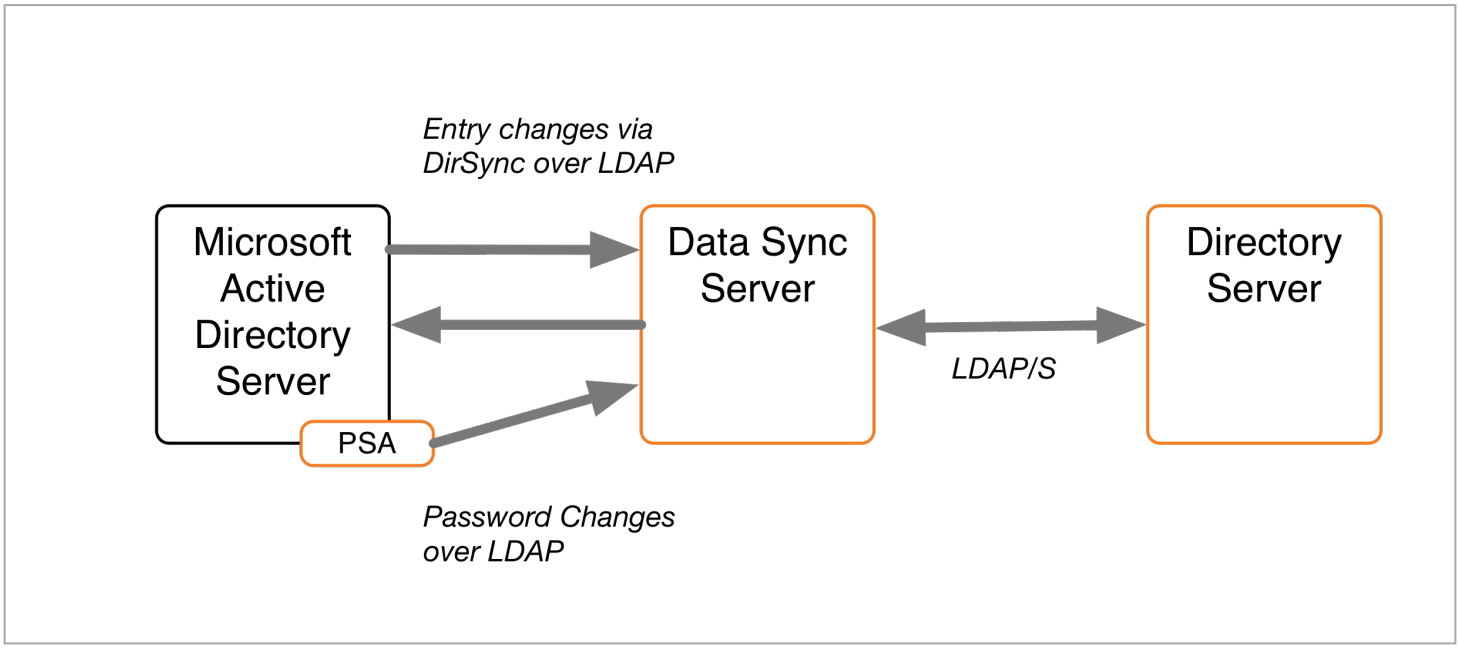
> **Note**
>
> The Password Sync Agent can't be pointed at multiple domain clusters.

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 AD, enable the Password Encryption component. See Configuring password encryption 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.

Important

If the Password Sync Agent is down for any length of time and misses a password change, the change won't be synced on recovery without either a new password change for the entry or the use of pass-through authentication.

The PSA is safe to leave running on a domain controller indefinitely. To stop synchronizing passwords, remove the `userPassword` attribute mapping in PingDataSync, 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 that 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.
- 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 and should point to the domain controller(s) on which the PSA will reside.
- You must add the certificate of the Active Directory Domain Controller (ADDC) to the certificate trust store of the PingDataSync server. To do this, export the certificate from the ADDC and then use the `manage-certificates trust-server-certificate` tool to import it into the trust store of the PingDataSync server.
- 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.

Installing the Password Sync Agent

About this task

The PingDirectory server distributes the PSA in zip file format with each PingDataSync package. The initial installation of the PSA requires a system restart. NOTE: The Password Sync Agent cannot be pointed at multiple domain clusters.

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 PingDataSyncs, such as `sync.host.com:636`. Do not add any prefixes to the host names.
4. Enter the Directory Manager distinguished name (DN) and password. This creates an ADSync user on PingDataSync.
5. Enter a password (secret key) for the ADSync user that will be used by the PSA when connecting to the PingDataSync 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 AD 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
```

Upgrading the Password Sync Agent

Before you begin

You must know which version of the PSA you're running. To check the version number, go to Control Panel → Programs → Programs and Features and find the row containing the Password Sync Agent item. The version number is displayed on that row.

> **Important**
>
> If you are running a version earlier than 4.7, you must uninstall the Password Sync Agent before upgrading because the installer has trouble locating the registry values from the older version. If you don't uninstall the older version of the PSA in this scenario, the upgraded PSA fails to start after the update.

About this task

You can upgrade the PSA by following these steps:

Steps

1. Extract the new version of the PSA.

   > **Important**
   >
   > Check the `Readme` file included with the PSA. If the header doesn't indicate a version, download version 4.7 or later before upgrading.

2. Follow the instructions for installing the PSA.

   > **Note**
   >
   > You don't need to restart the computer to complete this step. The core password filter DLL is already running under LSA. The update replaces the implementation binaries, which are encapsulated from the password filter DLL and can be updated dynamically.

Next steps

After you complete the installation, a dialog window might indicate that a restart is necessary due to configuration changes:

- If the logging service has restarted, you can safely ignore this message.

- If the logging service has not restarted, you should restart the computer to ensure that the PSA resumes syncing password changes.

Uninstalling the Password Sync Agent

About this task

You can use any of these options to uninstall the PSA from the AD system.

Steps

- Uninstall the PSA.

  Choose from:

  - Go to Control Panel → Programs → Programs and Features and double-click the Password Sync Agent item.
  - Run `setup.exe` for the installed version of the PSA and select Remove Password Sync Agent.

Result

The implementation DLL gets unloaded, and the database and log files are deleted. Only the binaries remain. The core password filter is still 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. After restarting the computer, you can delete the password filter and implementation binaries from the install folder.

> **Note**
>
> You must reboot the computer again before reinstalling the PSA.

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 supports high-scale, highly-available data synchronization between the directory servers and relational database management systems (RDBMS). Any database with a Java database connectivity (JDBC) 3.0 or later driver can be used.

Use the server SDK

Synchronizing Lightweight Directory Access Protocol (LDAP) data to or from a relational database requires creating a Java database connectivity (JDBC) Sync Source or Destination extension to act as an interface between PingDataSync 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.

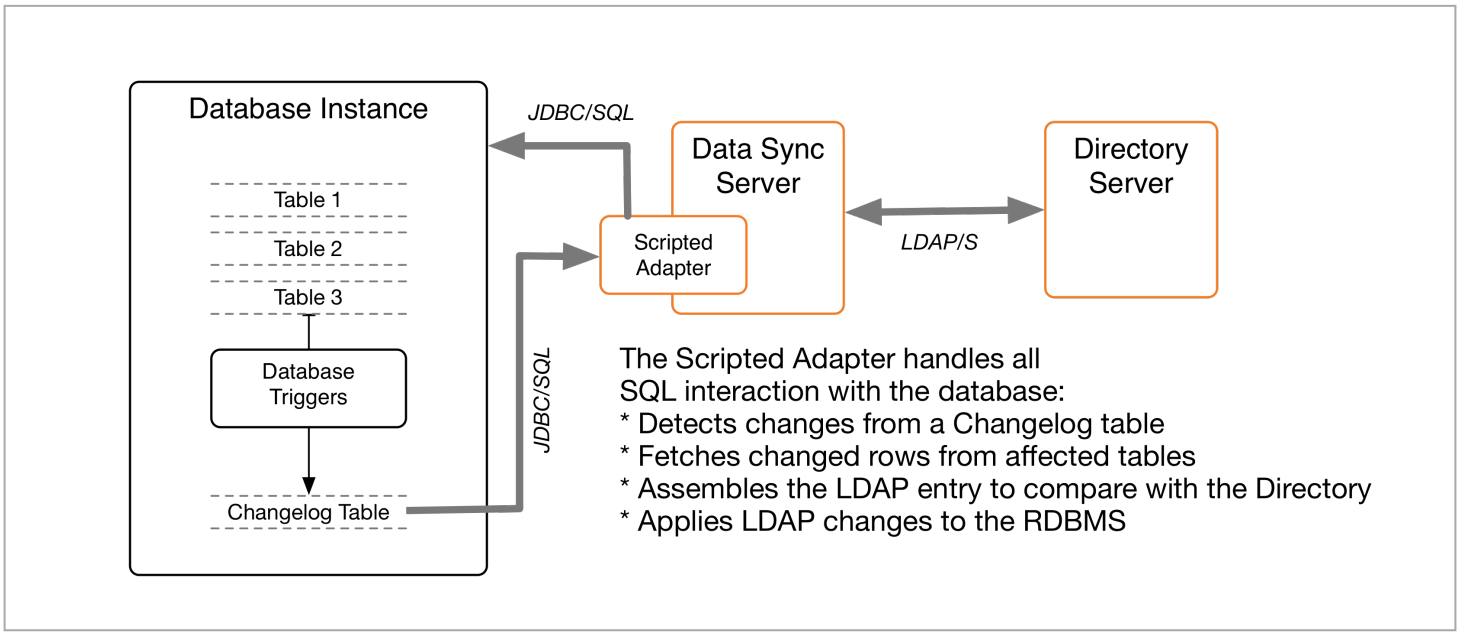
The Server SDK contains two abstract classes that correspond to how the database is used:

```
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 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. The following figure is a sample architecture.



RDBMS synchronization process

PingDataSync synchronizes data between a directory server and a relational database management system (RDBMS) with a Server SDK extension. PingDataSync provides multiple configuration options, such as advanced filtering (fractional and subtree), attribute and distinguished name (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.

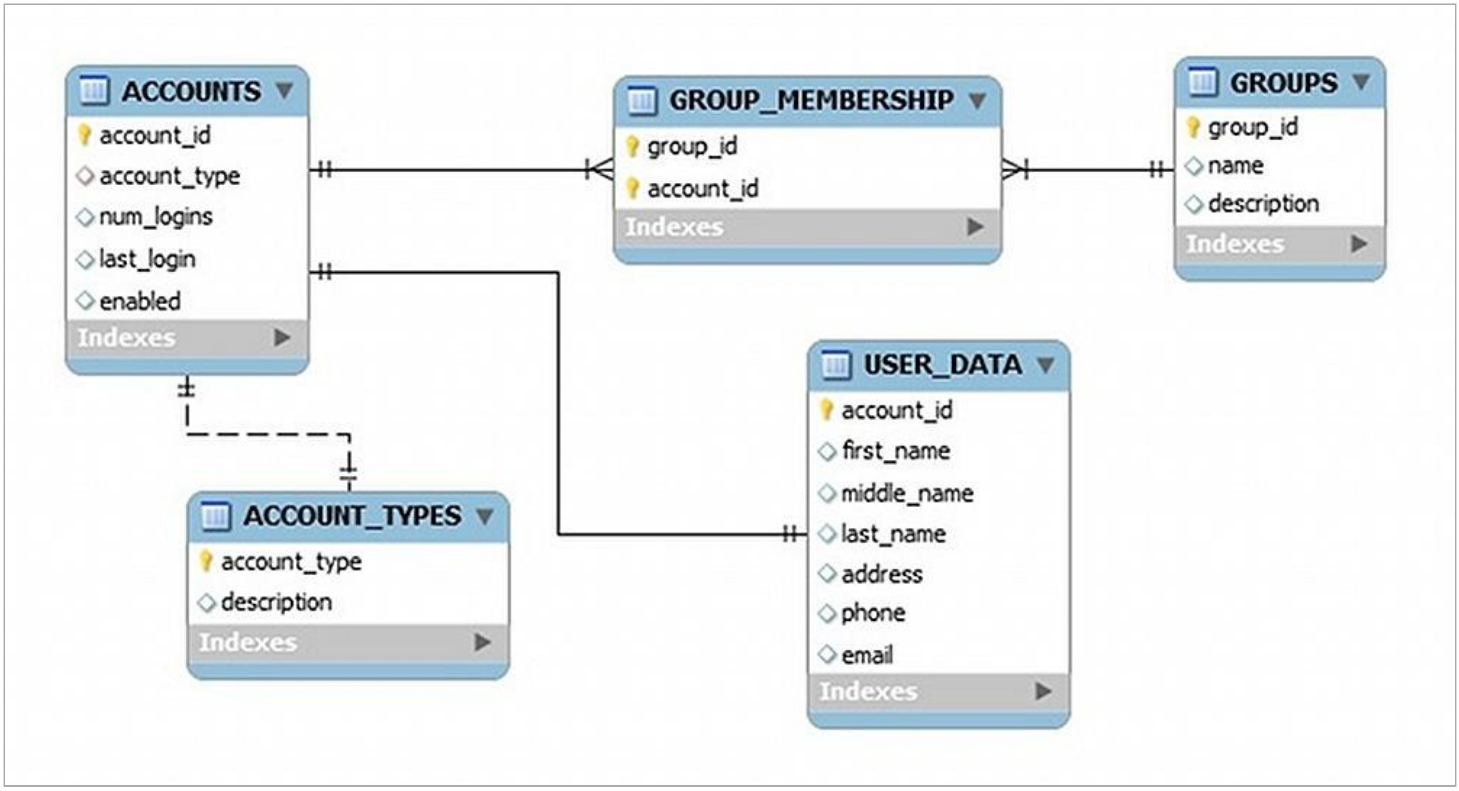
PingDataSync delegates the physical interaction with the database to a user-defined extension, which has full control of the SQL queries. The extension layer provides flexibility in how the mapping semantics between the Lightweight Directory Access Protocol (LDAP) environment and the relational database environment are defined. The connection management, pooling, retry logic, and other boilerplate code are handled internally by PingDataSync.

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 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 provides a DBSync example between two endpoints consisting of a PingDirectory server source and a relational database management system (RDBMS), which will be used in this topic.

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 Lightweight Directory Access Protocol (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 provided here:

Steps

1. Download the appropriate Java database connectivity (JDBC) driver to PingDataSync'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 and 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, distinguished name (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 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 Java database connectivity (JDBC) extension implementation must be written in Java or in 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 can 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` 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
- Lightweight Directory Access Protocol (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 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 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` 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 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 this script (through this method). If not, changes could be missed when PingDataSync 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 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` 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, see 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 is 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, 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 Lightweight Directory Access Protocol (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.

Create Specific DN Maps (optional)

If necessary, create a distinguished name (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 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 the PingDirectory server, 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 Configure the PingDirectory server backend for synchronizing deletes.

Attribute-Synchronization-Mode for DBSync

For MODIFY operations, PingDataSync 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 might 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 topics contain procedures that let you configure 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. You can configure Sync Pipes later using the `dsconfig` command.

Creating 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 distinguished name (DN) for the directory server, which is used as the base for Lightweight Directory Access Protocol (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 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 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 shown.

   ```
   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 Java database connectivity (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 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 PingDirectory 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 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.

Configuring 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` command to make changes to this configuration. Refer to Configuring PingDataSync for configuration options and details.

Considerations for synchronizing from a database source

When synchronizing from a database to a directory or relational database management system (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 Java database connectivity (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 Lightweight Directory Access Protocol (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 distinguished name (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.

See Using 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 Java database connectivity (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` command:

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 supports Kafka as a sync destination with a Ping Identity sync source, such as a PingDirectory server. In this scenario, the Kafka sync destination publishes changes in the 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 the 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 the 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 admin console to configure PingDataSync to synchronize changes to an Apache Kafka environment.

PingDataSync supports synchronization of single and multivalued attributes to Kafka. You can reuse existing Ping Identity sync sources that were created for other Sync Pipes.

> **Tip**
>
> To view an example configuration, see the file located at `<server-root>/config/sample-dsconfig-batch-files/reference-sync-pingdirectory-to-kafka.dsconfig`.
> To configure Kerberos authentication for a Kafka sync destination, supply the `producer-property` attribute with the appropriate values according to the Apache Kafka documentation.

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 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.

Obscuring sensitive producer property values

About this task

When configuring a PingDataSync Kafka producer, you might add producer properties that contain sensitive values such as keys or passwords. To prevent storing these sensitive values in plain text, you can use the `sensitive-kafka-producer-property` configuration property.

You create a `sensitive-kafka-producer-property` using the following required arguments:

`--property-name`

Specifies the name of the sensitive Kafka producer property.

`--set sensitive-producer-key:<key>`

Specifies the name of the valid property key that contains a sensitive value.

`--set sensitive-producer-value:<value>`

Specifies the sensitive value associated with the producer key.

Steps

- Create one or more sensitive Kafka producer properties using `dsconfig create-sensitive-kafka-producer-property`.

  Example:

```
$ bin/dsconfig create-sensitive-kafka-producer-property \
  --property-name saslConfig \
  --set "sensitive-producer-key:sasl.jaas.config" \
  --set "sensitive-producer-value:org.apache.kafka.common.security.scram.ScramLoginModule" \
    required username="username" password="password";
```

Result:

Perform an `ldapsearch` for the sensitive property:

```
ldapsearch --baseDN "cn=saslConfig,cn=Sensitive Kafka Producer Property,cn=config" "(objectclass=*)"
```

The sensitive value is now obscured.

```
dn: cn=saslConfig,cn=Sensitive Kafka Producer Property,cn=config
objectClass: top
objectClass: ds-cfg-sensitive-kafka-producer-property
cn: saslConfig
ds-cfg-sensitive-producer-key: sasl.jaas.config
ds-cfg-sensitive-producer-value: AADu9yRP8DyrLndvqqDzeQEK9aqqLvDBZZhgHAZbh+
+KgovN+kUthhyn9+1o9+AqExDmigO14YQnwakqOpTAB4LnbsvwBJos6PZzYlWMNjFNXsDtOUeBsFhVi/
nErPJT+cmQijC5P1EUsKWPvjDVauBe
```

> **Note**
>
> The `config-audit.log` file that contains the **`dsconfig`** change you made to create the sensitive property also obscures the value.

- (Optional) Delete one or more sensitive Kafka producer properties using **`dsconfig delete-sensitive-kafka-producer-property`**.

  Example:

```
$ bin/dsconfig delete-sensitive-kafka-producer-property \
--property-name saslConfig
```

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`

Configuration Object Type	Configuration Property	Kafka Configuration Property
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 distinguished name (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 message is a JavaScript Object Notation (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 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 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 maps the attributes and filters the results, any entries that are changed will use 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, see 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 can also synchronize data through a proxy server in both load-balanced and entry-balancing deployments.

> **Important**
>
> PingDirectoryProxy servers are the only supported endpoints.

The following topics detail a Sync-through-Proxy deployment and provide background information on how it works. Before setting up PingDataSync, review the PingDirectoryProxy Server Administration Guide.

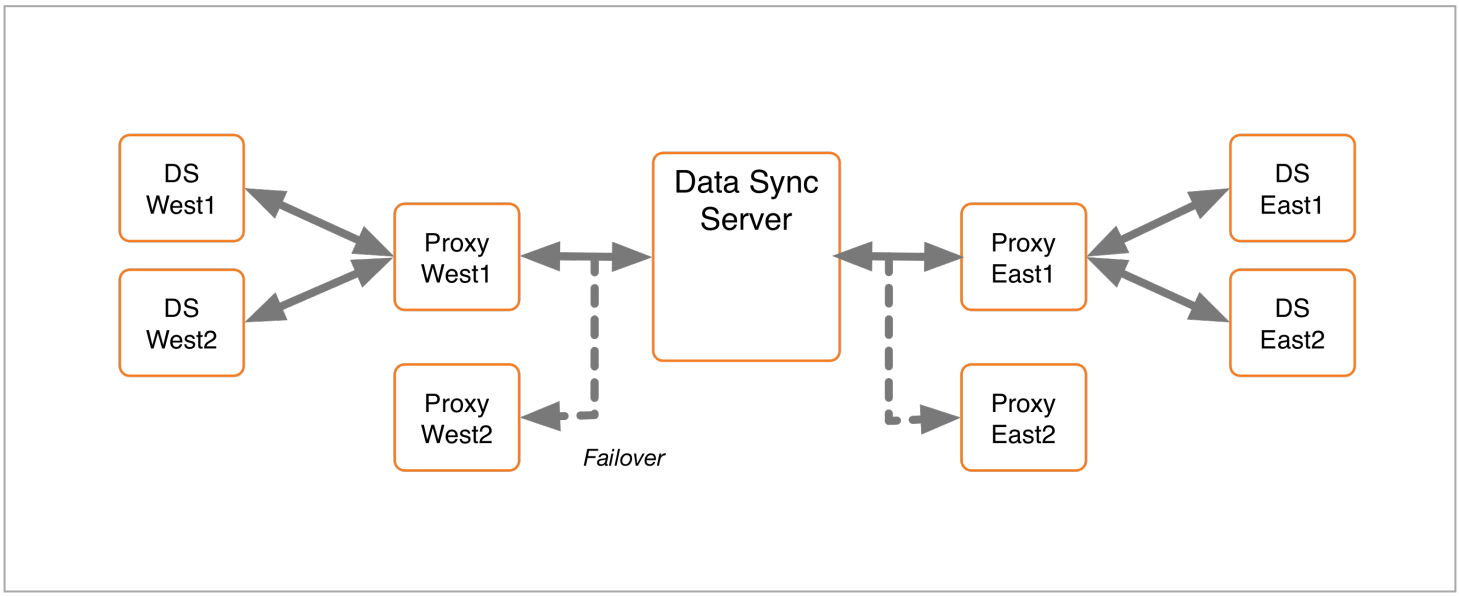
Synchronization through a PingDirectoryProxy server overview

To handle data synchronization through a PingDirectoryProxy server, PingDirectory servers have a `cn=changelog` state management system that supports a token-based API.

In a standard, non-proxied configuration, PingDataSync 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 passes the request through a proxy server to the backend set of directory servers. PingDataSync 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 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 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 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 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 because a shared area of data can exist above the entry-balancing base distinguished name (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:

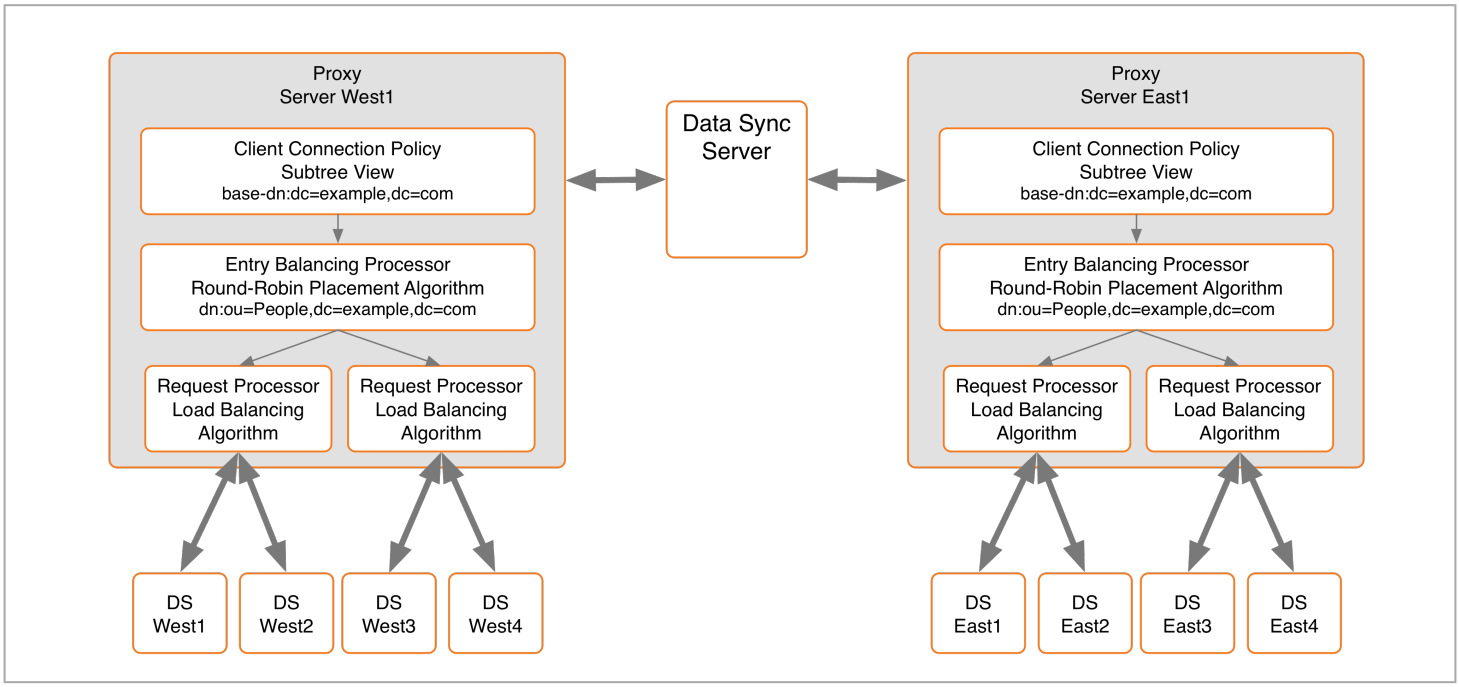
```
Proxy Token:
backendSetID 1: ds-token-1 {serverID A, changeNumber 1002, replicaState (5
(SA), 15(A)}
backendSetID 2: ds-token-2 {serverID B, changeNumber 2002, replicaState (10
(SB), 20(B)}
```

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 distinguished name (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 Lightweight Directory Access Protocol (LDAP) and failover servers are not installed or designated in PingDataSync.



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 PingDirectory 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 before 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 2 through 5 (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` command. 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 distinguished name (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

About this task

Configure PingDataSync 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, see Installing the PingDataSync server.

Steps

1. From the PingDataSync root directory, run the `create-sync-pipe-config` command.

```
$ 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 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 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.
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 and the PingDirectoryProxy servers. In addition to preparing the PingDirectoryProxy server, PingDataSync 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 before 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 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 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 cannot retrieve specific change numbers from the backend directory servers. Use `--change-sequence-number`, `--end-of-changelog`, or other available options.

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. Run the `realtime-sync start` command to start the Sync Pipe.

```
$ bin/realtime-sync start \
  --pipe-name "Ping Identity Proxy 1 to Ping Identity Proxy 2"
```

7. Monitor PingDataSync using the status commands and logs.

Index the LDAP changelog

About this task

The PingDirectory server supports 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 sends a get changelog batch request to the PingDirectoryProxy 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 Lightweight Directory Access Protocol (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, configure the `auto-map-source-attributes` property to specify the mappings for the attributes that need to be synchronized.

Result

PingDataSync 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 will use the get changelog batch request to retrieve changes from the Lightweight Directory Access Protocol (LDAP) changelog. This extended request can contain an optional set of selection criteria, which specifies changelog entries for a specific set of attributes.

PingDataSync takes the union of the source attributes from PingDataSync distinguished name (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 supports a notification synchronization mode that transmits change notifications on a source endpoint to third-party destination applications.

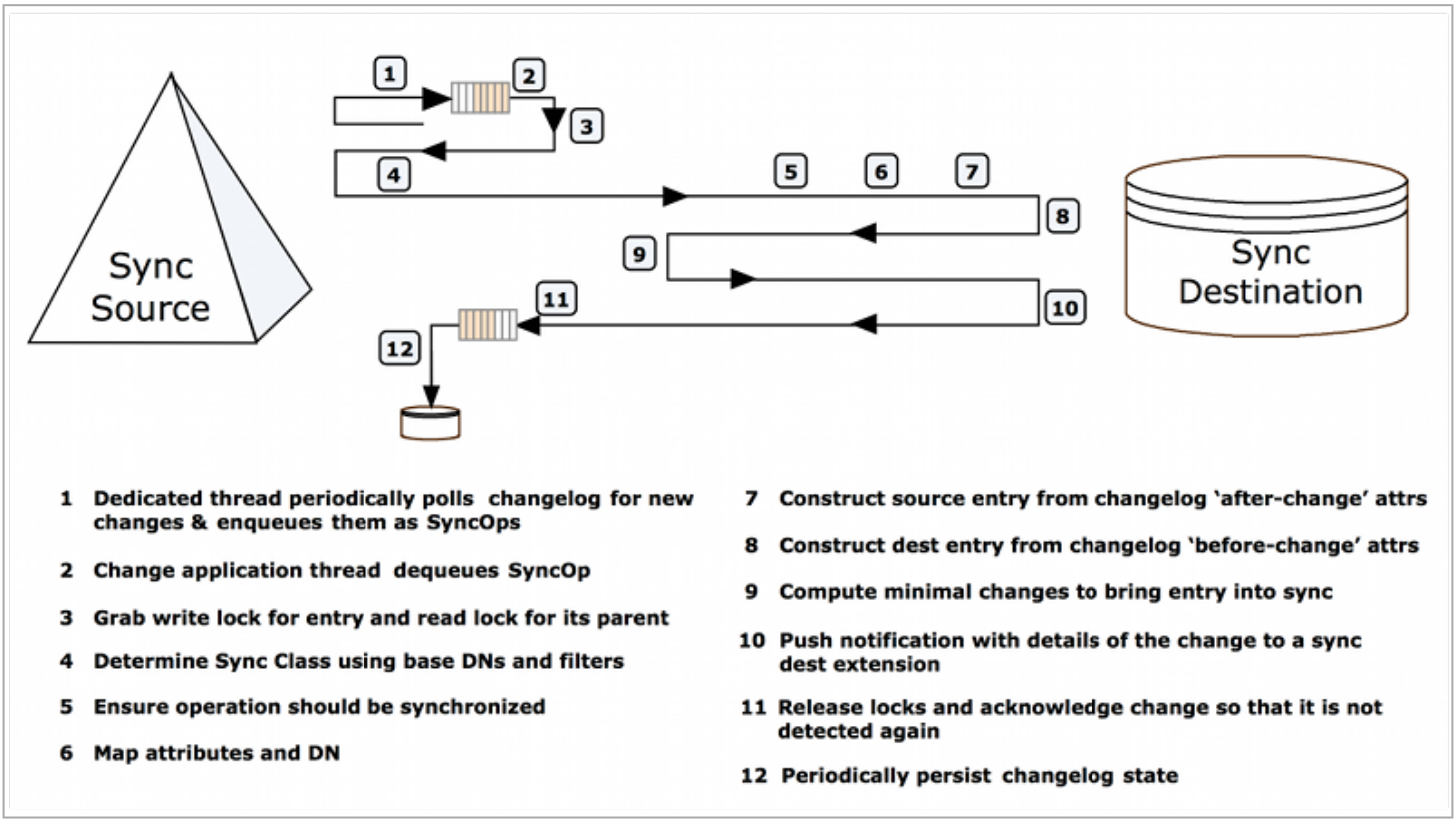
As with standard mode, notifications can be filtered based on the type of entry that was changed, the specific attributes that were changed, and the type of change (ADD, MODIFY, DELETE). PingDataSync can send a notification to arbitrary endpoints by using a custom server extension.

Notification mode overview

PingDataSync supports standard and notification synchronization modes. Notification mode polls the directory server's Lightweight Directory Access Protocol (LDAP) change log for changes on any entry but skips the fetch and compare phases of processing in 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 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 or PingDirectoryProxy server as the source endpoint.



PingDataSync 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 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` directory (for Groovy-based extensions) or the jar file in the `<server-root>/lib/extensions` directory (for Java-based extensions) before configuring PingDataSync 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. For detailed information and examples, see the LDAP SDK Javadoc.

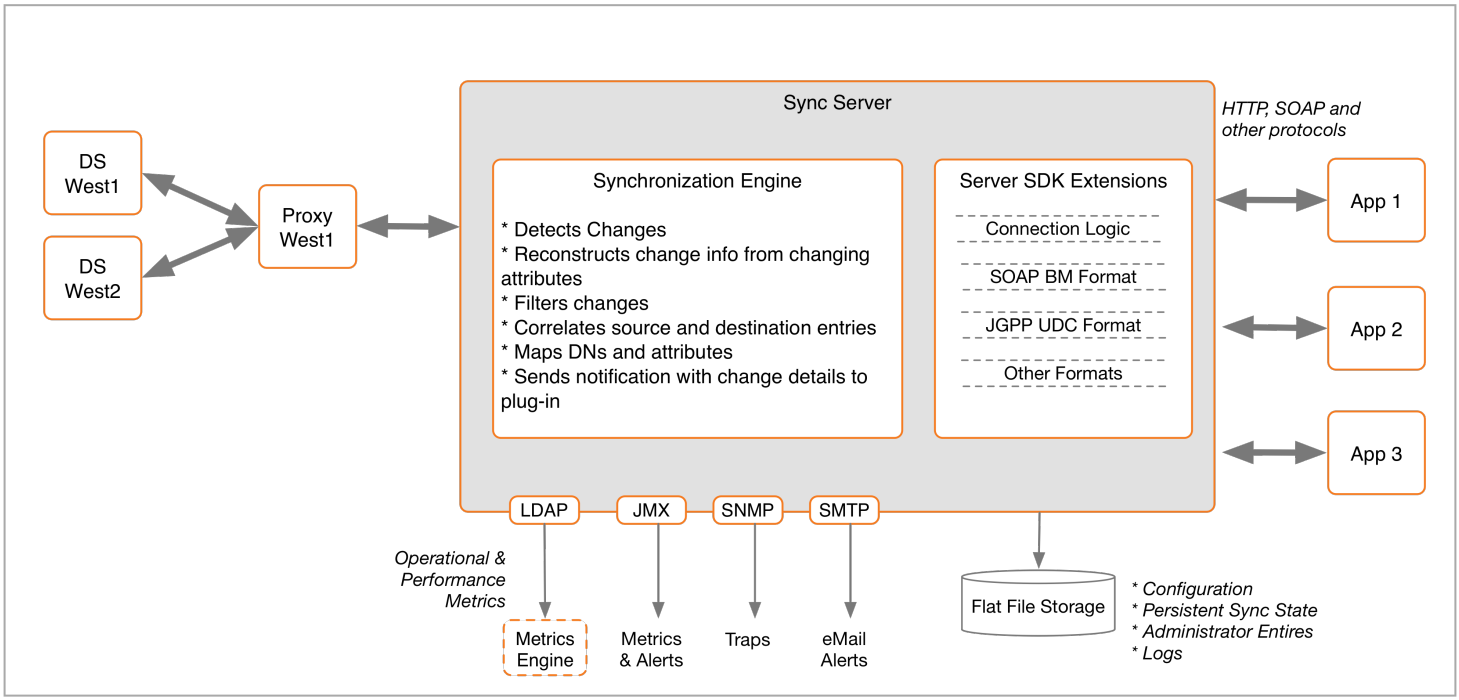
Notification mode architecture

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 detects the changes in the PingDirectory server's Lightweight Directory Access Protocol (LDAP) change log, filters the results specified in the Sync Classes, applies any distinguished name (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 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:

- PingDirectory Server and PingDirectoryProxy Server 3.0.5 or later
- A Sync destination of any type

You should not configure PingDataSync to interact with backend source servers through a load balancer. You can use PingDirectoryProxy for this configuration instead, assuming you are using PingDirectory servers as your backends.

> **Note**
>
> Although the PingDirectoryProxy server can front other vendor's directory servers, such as Active Directory (AD) and Sun DSEE, for processing LDAP operations, PingDataSync cannot synchronize changes from these sources through a proxy server. Synchronizing changes directly from AD and Sun DSEE cannot be done with notification mode.

Failover capabilities

For sync source failovers, configure replication between the PingDirectory 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 because 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 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 failovers, the secondary PingDataSync servers will be at or slightly behind the state where the primary server initiated a failover. Both primary and secondary PingDataSyncs track the last failed acknowledgement, so once the primary server fails over to a secondary server, the secondary server will not miss a change.

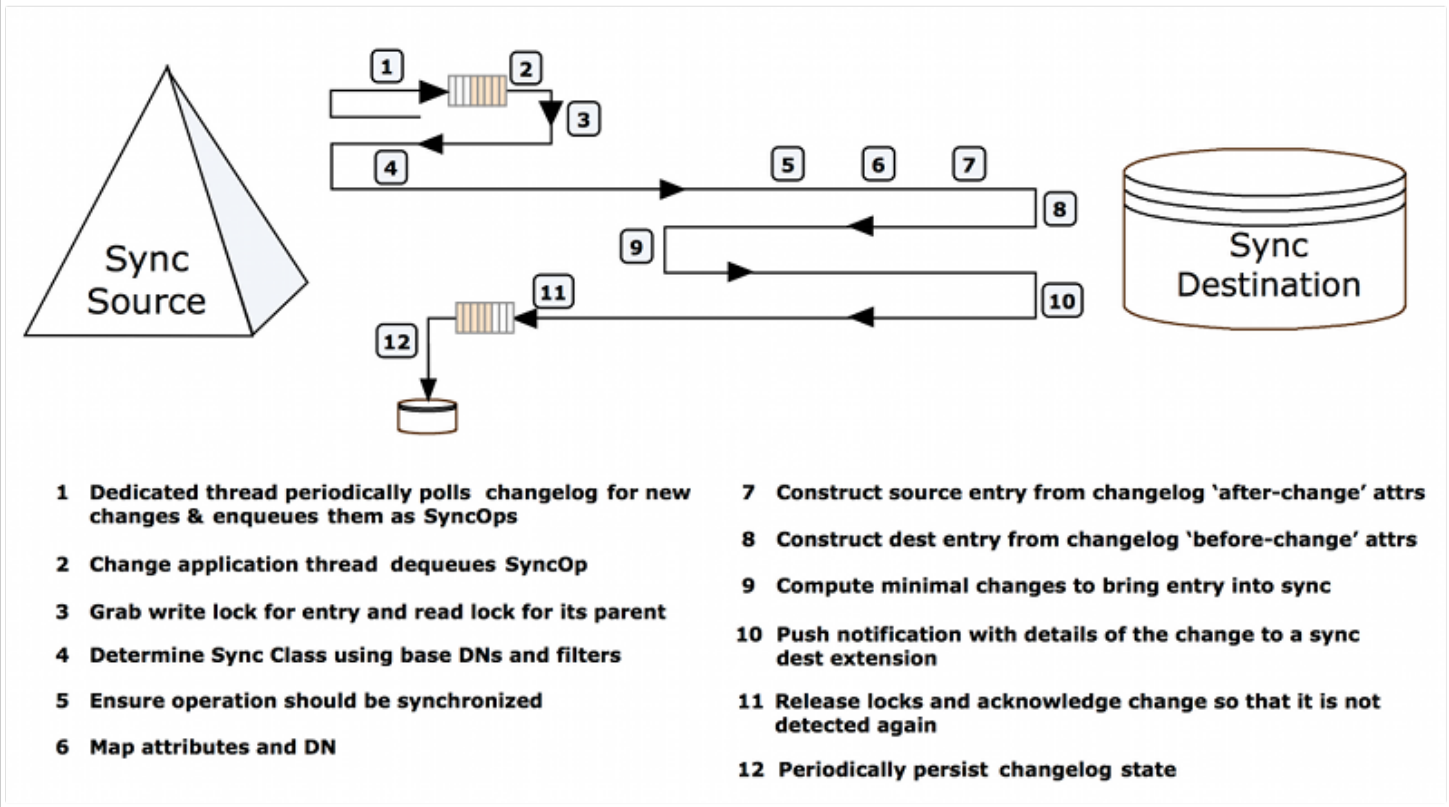
> **Note**
>
> If failover is a concern between PingDataSync servers, change the `sync-failover-polling-interval` property from 7500 ms to a smaller value. This will result in a quicker failover, but will marginally increase traffic between the two PingDataSync servers. The `sync-failover-connection-timeout` and `sync-failover-response-timeout` properties can also be updated to use different failover timeout durations. Use the **`dsconfig`** command to access the property on the Global Sync Configuration menu.

Notification sync pipe change flow

Multi-threaded Sync Pipes allow PingDataSync 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'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 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 requires 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. 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. For more information, see Configuring the PingDirectory server.

Normal Lightweight Directory Access Protocol (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 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 might 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
  ```

  In this multi-valued attribute definition, `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 without code changes to the core product. After 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. Learn more in Managing Server SDK extensions.

Review the Server SDK package

Review Server SDK documentation and examples before building and deploying 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, delete, and 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 Lightweight Directory Access Protocol (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 that can be used to generate log messages and/or alerts from the scripted layer: `logMessage()`, `sendAlert()`, `debugCaught()`, `debugError()`, `debugInfo()`, `debugThrown()`, `debugVerbose()`, and `debugWarning()`. These methods 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 might 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:

```
$ 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 the directory to a generic client application.

Configure the sync source

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 distinguished name (DN) for the directory server used for Lightweight Directory Access Protocol (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 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 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 to access the source endpoint server.

Configure the destination endpoint server

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 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. For more information see Sync user account.

Access control filtering on the sync pipe

PingDataSync 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 **`dsconfig`** commands.

The source server must be a PingDirectory server or a PingDirectoryProxy server that points to a PingDirectory 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

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 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 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 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 PingDirectory server and the PingDirectoryProxy server configured as SCIM servers using the SCIM extension.

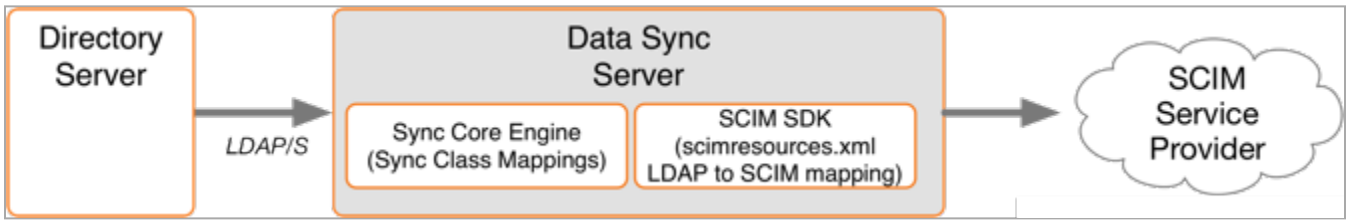
Synchronize with a SCIM sync destination overview

The System for Cross-domain Identity Management (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 can be configured to synchronize with SCIM service providers.

> **Note**
>
> Both the PingDirectory server and the PingDirectoryProxy server can be configured to be SCIM servers using the SCIM HTTP Servlet Extension.

PingDataSync is LDAP-centric and operates on Lightweight Directory Access Protocol (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 is LDAP-centric and typically at least one endpoint is an LDAP Directory server, this is not a strict requirement. For example, a Java database connectivity (JDBC) to SCIM sync pipe can be configured.

PingDataSync 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 an LDAP schema to and from SCIM.



> **Note**
>
> PingDataSync 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, and it 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 distinguished name (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 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 System for Cross-domain Identity Management (SCIM) using the `create-sync-pipe-config` utility and the `dsconfig` command. Configuring synchronization between an Lightweight Directory Access Protocol (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 distinguished name (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, 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 System for Cross-domain Identity Management (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 might not have the user name in distinguished name (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 can record which attributes were removed during a DELETE operation.

Perform the following steps to configure the Sync source:

Steps

1. Run the `dsconfig` command 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, 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 System for Cross-domain Identity Management (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 Lightweight Directory Access Protocol (LDAP) to System for Cross-domain Identity Management (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. After the source and destination endpoints have been configured, configure the Sync Pipe for LDAP to SCIM synchronization. Run `dsconfig` 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 `dsconfig` 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. Run `dsconfig` to set the base distinguished name (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 to include `inetOrgPerson` entries as user entries. The `destination-correlation-attributes` specifies LDAP attributes that allow PingDataSync 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 and the Lightweight Directory Access Protocol (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 set up communication between PingDataSync and the source server(s). The tool will prompt for the bind distinguished name (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 `realtime-sync` 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
```

Mapping the LDAP schema to the SCIM resource schema

The resources configuration file is used to define the System for Cross-domain Identity Management (SCIM) resource schema and its mapping to Lightweight Directory Access Protocol (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 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`. After this has been 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 distinguished name (DN)s 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 can 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 (in other words, 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 Mapping the LDAP schema to the 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 Mapping the LDAP schema to the SCIM resource schema.

<LDAPSearch> element

An `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 a destination correlation attribute, this setting is not used by PingDataSync.

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

An `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 can 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 System for Cross-domain Identity Management (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 Lightweight Directory Access Protocol (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.

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 might be usable as SCIM Sync Destinations.

Configuring synchronization to a SCIM 2.0 server

The PingDataSync server supports System for Cross-domain Identity Management (SCIM) 2.0 servers as a sync destination, which means that it is possible to push changes (including creates, modifies, and deletes) read from another source to a server using the SCIM 2.0 protocol described in RFC 7644.

> **Note**
>
> To view an example Active Directory to SCIM 2.0 configuration, see the file located at `<server-root>/config/sample-dsconfig-batch-files/reference-sync-activedirectory-to-scim2.dsconfig`.

> **Note**
>
> It is currently not possible to use a SCIM 2.0 server as a source, enabling changes made in that server to synchronize to other types of destinations.

The `create-sync-pipe-config` tool does not provide support for creating a sync pipe with a SCIM 2.0 sync destination, so you will need to create the necessary configuration manually using a tool like `dsconfig` or the admin console. This involves the following steps:

1. Creating a sync source to allow the PingDataSync server to pull the changes to be synchronized. See Configure the sync source.
2. Optionally configuring a changelog password decryption key in the PingDataSync server if the source is a PingDirectory server instance. See Configure the changelog password decryption key in the PingDataSync server (optional).
3. Configuring a SCIM 2.0 external server with the details necessary to connect and authenticate to the destination server. See Configure the SCIM 2.0 external server.
4. Configuring a set of SCIM 2.0 attribute mappings to describe how the PingDataSync server should map attributes from the internal LDAP representation of entries obtained from the sync source into a form that can be sent to the SCIM 2.0 server. See Configure SCIM 2.0 attribute mappings.
5. Configuring one or more SCIM 2.0 endpoint mappings to provide information about the endpoints to which changes will be synchronized in the SCIM 2.0 server, including the attribute mappings for attributes associated with each endpoint. See Configure SCIM 2.0 endpoint mappings.
6. Configuring a SCIM 2.0 sync destination, which references the SCIM 2.0 external server and endpoint mappings created in earlier steps, and provides additional configuration options. See Configure the SCIM 2.0 sync destination.
7. Configuring a sync pipe to associate the appropriate sync source with the SCIM 2.0 sync destination. See Configure a sync pipe.
8. Configuring one or more sync classes for that sync pipe to indicate how the PingDataSync server should treat various types of entries read from the source when pushing changes to the destination. See Configure sync classes.
9. Optionally using the `realtime-sync set-startpoint` command to set the startpoint at the end of the changelog if the source is a PingDirectory server instance. See Set the changelog startpoint for the sync source (optional).
10. Using the `resync` tool to perform an initial bulk synchronization of the content in the sync source into the SCIM 2.0 sync destination. See Perform an initial bulk synchronization with the `resync` command.

11. Using the `realtime-sync start` command to start synchronizing changes from the source into the SCIM 2.0 sync destination. See Start real-time synchronization.

Configure the sync source

The sync source describes the service from which entries and changes are read so that they can be synchronized to the sync destination.

The process for configuring a sync source varies based on the type of service that you use, such as an LDAP server or a relational database, so you should consult the appropriate documentation for the specific type of sync source that you want to use.

Currently, the `create-sync-pipe-config` tool does not offer support for the System for Cross-domain Identity Management (SCIM) 2.0 sync destination, so you might need to configure the sync source manually with a tool like `dsconfig` or the admin console. However, if you plan to synchronize from the desired source to another type of destination, and if that destination is one that is supported by the `create-sync-pipe-config` tool, then you can reuse the sync source created for that pipe.

If the sync source server is a PingDirectory server, then you can use the `prepare-endpoint-server` tool to make necessary changes to allow the PingDataSync server to interact with that directory server instance. This includes creating the account that the PingDataSync server uses to authenticate to the PingDirectory server and enabling the changelog to allow the PingDataSync server to retrieve information about changes processed in the PingDirectory server.

Running `prepare-endpoint-server --help` shows you the complete usage for the tool, but the following example demonstrates a sample usage:

```
bin/prepare-endpoint-server \
     --hostname ds-source.example.com \
     --port 636 \
     --useSSL \
     --trustStorePath config/truststore \
     --syncServerBindDN "cn=Sync User,cn=Root DNs,cn=config" \
     --syncServerBindPasswordFile sync-user-password.txt \
     --baseDN dc=example,dc=com \
     --isSource
```

In addition, if the source server is a PingDirectory server instance, then you should enable the Changelog Password Encryption plugin in that server to indicate that it should store an encrypted representation of clear-text passwords in the changelog along with their encoded form. See Configuring password encryption.

Doing this allows the PingDataSync server to retrieve those clear-text passwords so that they can be synchronized to the SCIM 2.0 sync destination. You can do this with a change like the following:

```
dsconfig set-plugin-prop \
     --plugin-name "Changelog Password Encryption" \
     --set enabled:true \
     --set changelog-password-encryption-key:<this-is-the-key-you-want-to-use>
```

Configure the changelog password decryption key in the PingDataSync server (optional)

If the sync source is a PingDirectory server instance that has the Changelog Password Encryption plugin enabled, then you need to configure the PingDataSync server with the same key so that it can decrypt those passwords.

You can do this with a change like the following:

```
dsconfig set-global-sync-configuration-prop \
     --set changelog-password-decryption-key:<this-is-the-key-you-want-to-use>
```

Configure the SCIM 2.0 external server

The System for Cross-domain Identity Management (SCIM) 2.0 external server configuration object provides the information that the PingDataSync server needs to connect and authenticate to the SCIM 2.0 service.

First, you must create an HTTP authentication method that allows the PingDataSync server to authenticate to the SCIM 2.0 server to authorize requests. In most cases, this authentication is an OAuth 2.0 bearer token, and you will likely want to obtain that token using the client credentials grant type. This allows you to provide a client ID and client secret to the OAuth authorization server to obtain a bearer token.

In this case, the client secret is sensitive information, so the PingDataSync server uses a passphrase provider to access it, which allows it to be obtained from a variety of sources, like an optionally encrypted file, Amazon Secrets Manager, Azure Key Vault, a CyberArk Conjur instance, or a HashiCorp Vault instance. For example:

```
dsconfig create-passphrase-provider \
     --provider-name "SCIMv2 Client Secret" \
     --type file-based \
     --set enabled:true \
     --set password-file:config/scimv2-client-secret.txt

dsconfig create-http-authorization-method \
     --method-name "SCIMv2 Authorization Method" \
     --type client-credentials-bearer-token \
     --set enabled:true \
     --set oauth-server-token-endpoint-url:https://oauth.example.com/as/token \
     --set hostname-verification-method:strict \
     --set oauth-client-id:this-is-the-client-id \
     --set "oauth-client-secret-passphrase-provider:SCIMv2 Client Secret" \
     --set request-method:get \
     --set credentials-submission-method:basic-authorization \
     --set "maximum-token-lifetime:1 h"
```

The SCIM 2.0 external server configuration offers the following properties:

`scim-service-url`

> The base URL to the SCIM 2.0 service to be used. This should not include any endpoint name because that will be appended through the endpoint mapping. This is required.

`key-manager-provider`

A key manager provider to use during SSL negotiation with the SCIM 2.0 server. This is optional, and it will likely only be used if the PingDataSync server needs to supply a client certificate to the SCIM 2.0 server.

`ssl-cert-nickname`

The nickname (alias) of the client certificate to present to the SCIM 2.0 server. This is only needed if a `key-manager-provider` is specified and only if the associated key store has multiple certificates that could be used.

`trust-manager-provider`

A trust manager provider to use to determine whether to trust the certificate chain presented by the SCIM 2.0 server during Secure Sockets Layer (SSL) negotiation. This is optional, and if you don't specify it, then the PingDataSync server will rely primarily on the Java Virtual Machine (JVM)'s default set of trusted issuers. If the SCIM 2.0 server is using a certificate signed by one of those trusted issuers, then you can leave this property unset.

`hostname-verification-method`

Indicates whether the PingDataSync server should verify that the certificate presented by the SCIM 2.0 server is appropriate for the intended address. A value of `strict`, which is the default, indicates that the connection should only be established if the certificate has a subject alternative name extension with a value that matches the address provided in the `scim-service-url` property (or if the certificate does not have a subject alternative name extension, then it falls back to using the `CN` attribute of the certificate subject). A value of `allow-all` indicates that the PingDataSync server should not attempt to confirm that the certificate was issued for the intended server.

`http-authorization-method`

The HTTP authorization method that the PingDataSync server should use to authenticate to and authorize requests in the SCIM 2.0 server. This is required.

`response-timeout`

The maximum length of time that the PingDataSync server should wait for a response from the SCIM 2.0 server when issuing requests. If this is not specified, a default of 10 seconds is used.

`client-reconnect-interval`

The maximum length of time that a SCIM 2.0 client instance will be used before a new one is created, which might potentially include obtaining new credentials. If the client credentials grant HTTP authorization method is used and the OAuth authorization server specified an expiration time for the bearer token that it issued, then the actual reconnect interval is based on the lesser of the two values. If this is not specified, and if the HTTP authorization method does not indicate a maximum lifetime for its credentials, then the same SCIM 2.0 client instance is used indefinitely.

> **Note**
>
> The server will automatically try to refresh the credentials if the SCIM 2.0 service returns a 401 (unauthorized) error in response to any request.

For example, you can use the following change to configure a SCIM 2.0 external server:

```
dsconfig create-external-server \
     --server-name "SCIMv2 Server" \
     --type scim2 \
     --set scim-service-url:https://scim2.example.com/scim/v2 \
     --set "http-authorization-method:SCIMv2 Authorization Method"
```

Configure SCIM 2.0 attribute mappings

System for Cross-domain Identity Management (SCIM) 2.0 attribute mappings are used to construct the SCIM representations of attributes from the information contained in the internal mapped Lightweight Directory Access Protocol (LDAP) representation of an entry that SCIM uses.

Even though the PingDataSync server can perform its own attribute mapping on data read from the source server (as configured in the sync class), this isn't sufficient on its own to interact with a SCIM 2.0 server.

In some cases, it might be necessary to transform values in a way that's not easily accomplished with the existing attribute mapping mechanism that the PingDataSync server provides, for example, from a generalized time representation that is typically used for timestamps in LDAP entries to the ISO 8601 format described in RFC 3339 that SCIM 2.0 servers typically use.

The existing attribute mapping can also be inconvenient for generating SCIM 2.0 complex attributes.

Each SCIM 2.0 attribute mapping is associated with a syntax. The server supports the following types of SCIM 2.0 attribute mappings.

SCIM 2.0 attribute mappings	Description
String	Used for SCIM 2.0 attributes represented as JSON strings. See String SCIM 2.0 attribute mappings.
Number	Used for SCIM 2.0 attributes represented as JSON numbers. See Number SCIM 2.0 attribute mappings.
Boolean	Used for SCIM 2.0 attributes represented as JSON Boolean values. See Boolean SCIM 2.0 attribute mappings.
DateTime	Used for SCIM 2.0 attributes in which the LDAP representation is a timestamp in the generalized time format, and the SCIM 2.0 representation uses the format described in RFC 3339. See DateTime SCIM 2.0 attribute mappings.
Postal Address	Used for SCIM 2.0 attributes in which the LDAP representation is a string that represents multiple lines using the dollar sign character ($) as the line delimiter, and the SCIM 2.0 representation uses newline characters (\n) to separate the lines. See Postal address SCIM 2.0 attribute mappings.
Composed Complex	Used for SCIM 2.0 complex attributes in which the sub-attributes are defined with other SCIM 2.0 attribute mappings. See Composed complex SCIM 2.0 attribute mappings

SCIM 2.0 attribute mappings	Description
JSON-Formatted Complex	Used for SCIM 2.0 complex attributes in which the JSON objects that represent the complex attributes are read from a specified attribute in the mapped LDAP representation of the entry. See JSON-formatted complex SCIM 2.0 attribute mappings.

Each of these will be described in more detail in its own section, but all types of attribute mappings share the following configuration properties:

`scim-attribute-name`

The name of the attribute as it appears in SCIM 2.0 entries. This is required.

`attribute-usage`

The ways in which the associated attribute is used. It must have at least one of the following values:

- `fetch` – The attribute is to be retrieved from the SCIM 2.0 server when fetching entries.
- `create-during-realtime-sync` – The attribute is to be included when creating a new entry in the SCIM 2.0 server during real-time synchronization.
- `create-during-resync` – The attribute is to be included when creating a new entry in the SCIM 2.0 server during resync processing.
- `update-during-realtime-sync` – The attribute is to be updated if necessary when applying changes to existing entries in the SCIM 2.0 server during real-time synchronization.
- `update-during-resync` – The attribute is to be updated if necessary when applying changes to existing entries in the SCIM 2.0 server during resync processing.

`failed-mapping-behavior`

Indicates how the PingDataSync server should handle cases in which an error occurs when trying to map a value between the LDAP and SCIM 2.0 representations of an entry, such as if the value doesn't conform to the expected syntax. The value for this property can be one of the following:

- `reject` – Indicates that the associated synchronization operation should be rejected. This is the default behavior that will be used if no value is specified.
- `ignore-entire-attribute` – Indicates that the associated synchronization operation should proceed as if the entire attribute had been omitted, even if it's a multivalued attribute in which other values can be successfully mapped.
- `ignore-individual-values` – Indicates that the associated synchronization operation should proceed as if any unmappable values had been omitted. For single-valued attributes, or for multivalued attributes in which none of the values can be successfully mapped, this behavior will be the same as ignore-entire-attribute. For multivalued attributes in which at least one value can be successfully mapped, only the unmappable values will be omitted.

always-patch-with-replace

Indicates whether to always use the `replace` operation type when updating values of this attribute with a SCIM 2.0 PATCH operation (this property does not apply when replacing the entire entry with a PUT operation). By default, this property has a value of `false`, to indicate that it should attempt to update values for the attribute with the `add` and `remove` operation types when possible. However, for some types of attributes (for example, if the SCIM 2.0 server allows you to update passwords but not retrieve them, or if it only allows retrieving them in an encoded form that does not match the form in which it was obtained from the source server), it might be necessary to always use the `replace` operation type.

Just as it might be necessary to change the value of the `always-patch-with-replace` property when synchronizing passwords, you might also need to treat the `attribute-usage` property specially for passwords as well. This is especially true in cases where the clear-text password is available during real-time synchronization, but not during a resync.

This can occur, for example, when synchronizing from a PingDirectory server with the Changelog Password Encryption Plugin enabled because the plugin includes an encrypted representation of the clear-text password in the changelog record for operations in which a clear-text password was provided, but the clear-text password is not available during resync processing.

In such cases, you might need to omit the `create-during-resync` and `update-during-resync` behaviors if the SCIM 2.0 server cannot handle passwords in their encoded form.

String SCIM 2.0 attribute mappings

String System for Cross-domain Identity Management (SCIM) 2.0 attribute mappings can be used for cases in which the SCIM attribute has a value that is a single JavaScript Object Notation (JSON) string or an array of JSON strings. The value will not be transformed in any way when converting between the Lightweight Directory Access Protocol (LDAP) and SCIM representations of an entry.

Additional configuration properties that are available for string SCIM 2.0 attribute mappings include:

ldap-attribute-name

The name of the LDAP attribute (in the mapped representation of the source entry generated by the sync class) whose values will be used as the values of the SCIM 2.0 attribute. This is required.

single-valued

Indicates whether the SCIM 2.0 attribute is single-valued or multivalued. If this is `true`, then the SCIM 2.0 representation of the attribute value will be as a JSON string. If this is `false`, then the SCIM 2.0 representation of the attribute value will be as a JSON array of strings.

You can use the following example configuration change to create a string SCIM 2.0 attribute mapping:

```
dsconfig create-scim2-attribute-mapping \
     --mapping-name userName \
     --type string \
     --set scim-attribute-name:userName \
     --set ldap-attribute-name:uid \
     --set attribute-usage:fetch \
     --set attribute-usage:create-during-realtime-sync \
     --set attribute-usage:create-during-resync \
     --set attribute-usage:update-during-realtime-sync \
     --set attribute-usage:update-during-resync \
     --set single-valued:true
```

Number SCIM 2.0 attribute mappings

You can use number System for Cross-domain Identity Management (SCIM) 2.0 attribute mappings for cases in which the SCIM attribute has a value that is a single JavaScript Object Notation (JSON) number (whether integer or floating-point) or an array of JSON numbers. The value will not be transformed in any way when converting between the Lightweight Directory Access Protocol (LDAP) and SCIM representations of an entry.

Additional configuration properties that are available for number SCIM 2.0 attribute mappings include:

`ldap-attribute-name`

The name of the LDAP attribute (in the mapped representation of the source entry generated by the sync class) whose values will be used as the values of the SCIM 2.0 attribute. This is required, and the LDAP attribute must have values that can be parsed as JSON numbers.

`single-valued`

Indicates whether the SCIM 2.0 attribute is single-valued or multivalued. If this is `true`, then the SCIM 2.0 representation of the attribute value will be as a JSON string. If this is `false`, then the SCIM 2.0 representation of the attribute value will be as a JSON array of strings.

You can use the following example configuration change to create a number SCIM 2.0 attribute mapping:

```
dsconfig create-scim2-attribute-mapping \
     --mapping-name "Example Numeric Value" \
     --type number \
     --set scim-attribute-name:exampleSCIMAttributeName \
     --set ldap-attribute-name:exampleLDAPAttributeName \
     --set attribute-usage:fetch \
     --set attribute-usage:create-during-realtime-sync \
     --set attribute-usage:create-during-resync \
     --set attribute-usage:update-during-realtime-sync \
     --set attribute-usage:update-during-resync \
     --set single-valued:true
```

Boolean SCIM 2.0 attribute mappings

Boolean System for Cross-domain Identity Management (SCIM) 2.0 attribute mappings can be used for cases in which the SCIM attribute has a value that is a single JavaScript Object Notation (JSON) Boolean value.

Additional configuration properties that are available for Boolean SCIM 2.0 attribute mappings include:

`ldap-attribute-name`

The name of the Lightweight Directory Access Protocol (LDAP) attribute (in the mapped representation of the source entry generated by the sync class) whose value will be used as the value of the SCIM 2.0 attribute. This is required, and the LDAP attribute must have a value of either `true` or `false`.

> **Note**
>
> This SCIM 2.0 attribute mapping type does not support multivalued attributes because it doesn't make sense for a Boolean attribute to have multiple values.

You can use the following example configuration change to create a Boolean SCIM 2.0 attribute mapping:

```
dsconfig create-scim2-attribute-mapping \
     --mapping-name "Account Enabled" \
     --type boolean \
     --set scim-attribute-name:active \
     --set ldap-attribute-name:accountEnabled \
     --set attribute-usage:fetch \
     --set attribute-usage:create-during-realtime-sync \
     --set attribute-usage:create-during-resync \
     --set attribute-usage:update-during-realtime-sync \
     --set attribute-usage:update-during-resync \
     --set single-valued:true
```

DateTime SCIM 2.0 attribute mappings

DateTime System for Cross-domain Identity Management (SCIM) 2.0 attribute mappings can be used for cases in which the SCIM attribute has a value that is a JavaScript Object Notation (JSON) string or an array of JSON strings representing timestamps in the ISO 8601 format described in RFC 3339.

The corresponding LDAP attribute has values in the generalized time format. The values will be transformed as appropriate when converting between the LDAP and SCIM 2.0 representations.

Additional configuration properties that are available for DateTime SCIM 2.0 attribute mappings include:

`ldap-attribute-name`

The name of the LDAP attribute (in the mapped representation of the source entry generated by the sync class) whose values will be used as the values of the SCIM 2.0 attribute. This is required, and the LDAP attribute must have values that can be parsed using the generalized time syntax.

single-valued

Indicates whether the SCIM 2.0 attribute is single-valued or multivalued. If this is `true`, then the SCIM 2.0 representation of the attribute value will be as a JSON string. If this is `false`, then the SCIM 2.0 representation of the attribute value will be as a JSON array of strings.

You can use the following example configuration change to create a DateTime SCIM 2.0 attribute mapping:

```
dsconfig create-scim2-attribute-mapping \
     --mapping-name "Create Time" \
     --type date-time \
     --set scim-attribute-name:created \
     --set ldap-attribute-name:createTimestamp \
     --set attribute-usage:fetch \
     --set attribute-usage:create-during-realtime-sync \
     --set attribute-usage:create-during-resync \
     --set single-valued:true
```

Postal address SCIM 2.0 attribute mappings

Postal address System for Cross-domain Identity Management (SCIM) 2.0 attribute mappings can be used for cases in which an attribute represents a postal address in a multi-line format.

In this case, the SCIM 2.0 representation of the address will use the newline character (\n) to separate lines, but the Lightweight Directory Access Protocol (LDAP) representation uses the dollar sign character ($) to separate lines. The values will be transformed as appropriate when converting between the LDAP and SCIM 2.0 representations.

Additional configuration properties that are available for postal address SCIM 2.0 attribute mappings include:

ldap-attribute-name

The name of the LDAP attribute (in the mapped representation of the source entry generated by the sync class) whose values will be used as the values of the SCIM 2.0 attribute. This is required.

single-valued

Indicates whether the SCIM 2.0 attribute is single-valued or multivalued. If this is `true`, then the SCIM 2.0 representation of the attribute value will be as a JavaScript Object Notation (JSON) string. If this is `false`, then the SCIM 2.0 representation of the attribute value will be as a JSON array of strings.

You can use the following example configuration change to create a postal address SCIM 2.0 attribute mapping:

```
dsconfig create-scim2-attribute-mapping \
     --mapping-name "Postal Address" \
     --type postal-address \
     --set scim-attribute-name:formatted \
     --set ldap-attribute-name:postalAddress \
     --set attribute-usage:fetch \
     --set attribute-usage:create-during-realtime-sync \
     --set attribute-usage:create-during-resync \
     --set attribute-usage:update-during-realtime-sync \
     --set attribute-usage:update-during-resync \
     --set single-valued:true
```

Dot notation support for SCIM 2.0 sub-attributes

There is a known issue with using dot notation for SCIM 2.0 operations on sub-attributes within an entry.

The PingDirectory server can't properly interpret the dot notation that defines sub-attributes in the SCIM 2.0 standard. To perform the desired operation, modify your SCIM statements by defining any sub-attributes as child objects of the parent attribute.

For example, when updating complex or extended SCIM attributes, don't use these formulations, which rely on dot notation to define the sub-attribute:

```
\\ Incorrect complex attribute example
        {
          "schemas":[
            "urn:ietf:params:scim:api:messages:2.0:PatchOp"
          ],
          "Operations":[
            {
              "op":"replace",
              "value":{
                "name.formatted":"test1 test2",
                "name.familyName":"test1"
              }
            }
          ]
        }
```

```
\\ Incorrect extended attribute example
{
  "schemas":[
        "urn:ietf:params:scim:api:messages:2.0:PatchOp"
  ],
  "Operations":[
        {
          "op":"replace",
          "value":{
                "urn:ietf:params:scim:schemas:extension:myown:2.0:User.lastname":"test3",
                "urn:ietf:params:scim:schemas:extension:myown:2.0:User.section":"test4"
          }
        }
  ]
}
```

Use the following formulations instead:

```
\\ Correct complex attribute example
        {
          "schemas":[
            "urn:ietf:params:scim:api:messages:2.0:PatchOp"
          ],
          "Operations":[
            {
              "op":"replace",
              "value":{
                "name":{
                  "formatted":"test1 test2",
                  "familyName":"test1"
                }
              }
            }
          ]
        }
```

```
\\ Correct extended attribute example
        {
          "schemas":[
                "urn:ietf:params:scim:api:messages:2.0:PatchOp"
          ],
          "Operations":[
            {
              "op":"replace",
              "value":{
                "urn:ietf:params:scim:schemas:extension:myown:2.0:User":{
                  "lastname":"test3",
                  "section":"test4"
                }
              }
            }
          ]
        }
```

Composed complex SCIM 2.0 attribute mappings

Composed complex System for Cross-domain Identity Management (SCIM) 2.0 attribute mappings can be used to create a single-valued complex attribute in which the sub-attributes are created from other SCIM 2.0 attribute mappings.

For example, the `name` complex attribute described in RFC 7643 section 8.7.1 can have sub-attributes, such as `formatted`, `familyName`, and `givenName`, that might correspond to the `cn`, `sn`, and `givenName` Lightweight Directory Access Protocol (LDAP) attributes. If you had SCIM 2.0 attribute mappings defined for each of those attributes, then you could use a composed complex SCIM 2.0 attribute mapping that uses those mappings to construct an appropriate value for the name complex attribute.

> **Note**
>
> Because the order in which values are presented in multivalued LDAP attributes is not considered significant, you can only use composed complex SCIM 2.0 attribute mappings to generate single-valued complex attributes. If you need a multivalued complex attribute, use the JavaScript Object Notation (JSON)-formatted complex SCIM 2.0 attribute mapping type.

Additional configuration properties that are available for composed complex SCIM 2.0 attribute mappings include:

`sub-attribute-mapping`

A reference to one or more attribute mappings for the sub-attributes to include in the complex attribute. This is required.

You can use the following example configuration change to create a composed complex SCIM 2.0 attribute mapping:

```
dsconfig create-scim2-attribute-mapping \
     --mapping-name "name Complex Attribute" \
     --type composed-complex \
     --set scim-attribute-name:name\
     --set attribute-usage:fetch \
     --set attribute-usage:create-during-realtime-sync \
     --set attribute-usage:create-during-resync
     --set attribute-usage:update-during-realtime-sync \
     --set attribute-usage:update-during-resync \
     --set "sub-attribute-mapping:Formatted Name" \
     --set "sub-attribute-mapping:First Name" \
     --set "sub-attribute-mapping:Last Name"
```

JSON-formatted complex SCIM 2.0 attribute mappings

JavaScript Object Notation (JSON)-formatted complex System for Cross-domain Identity Management (SCIM) 2.0 attribute mappings can be used to create complex SCIM attributes from Lightweight Directory Access Protocol (LDAP) attributes whose values are the string representations of JSON objects.

Unless the data is already stored in that format in the sync source, then it could be more complicated to produce this format (possibly requiring sync attribute mappings or even a sync destination plugin), but it does provide the ability to generate multivalued complex attributes.

Additional configuration properties that are available for JSON-formatted complex SCIM 2.0 attribute mappings include:

`ldap-attribute-name`

The name of the LDAP attribute (in the mapped representation of the source entry generated by the sync class) whose values will be used as the values of the SCIM 2.0 attribute. This is required, and the LDAP attribute must have values that can be parsed as JSON objects.

`single-valued`

Indicates whether the SCIM 2.0 attribute is single-valued or multivalued. If this is `true`, then the SCIM 2.0 representation of the attribute value will be as a JSON string. If this is `false`, then the SCIM 2.0 representation of the attribute value will be as a JSON array of strings.

You can use the following example configuration change to create a JSON-formatted complex SCIM 2.0 attribute mapping:

```
dsconfig create-scim2-attribute-mapping \
     --mapping-name "Email Addresses" \
     --type json-formatted-complex \
     --set scim-attribute-name:emails \
     --set ldap-attribute-name:ubidEmailJSON \
     --set attribute-usage:fetch \
     --set attribute-usage:create-during-realtime-sync \
     --set attribute-usage:create-during-resync \
     --set attribute-usage:update-during-realtime-sync \
     --set attribute-usage:update-during-resync \
     --set single-valued:false
```

Custom SCIM 2.0 attribute mappings for extended schemas

You can map custom attributes defined in extended schemas to System for Cross-domain Identity Management (SCIM) 2.0 sync destinations.

Consider the following example JSON that creates `testUser` using SCIM 2.0 and includes the custom attributes `workAnniversary` and `employeeAge` from an extended schema:

```
{
"schemas": [
"urn:ietf:params:scim:schemas:extension:gluu:2.0:User",
"urn:ietf:params:scim:schemas:core:2.0:User"
],
"id": "7e929a2d-18d3-462f-8c32-653a9ed170e2",
"meta": {
"resourceType": "User",
"created": "2022-12-07T03:33:45.469Z",
"lastModified": "2022-12-07T03:34:45.830Z",
"location": "https://rhel8/identity/restv1/scim/v2/Users/7e929a2d-18d3-462f-8c32-653a9ed170e2"
},
"userName": "testUser",
"name": {
"familyName": "User",
"givenName": "Test",
"formatted": "Test User"
},
"active": true,
"displayName": "Test User",
"urn:ietf:params:scim:schemas:extension:gluu:2.0:User": {
"workAnniversary": "1994-12-16T10:32:00Z",
"employeeAge": 55
}
}
```

To map custom attributes for synchronization with the SCIM 2.0 destination, you must:

- Create a composed complex attribute mapping with the custom schema URN
- Define any custom attributes associated with the custom schema as sub-attributes of the complex attribute

For example, to map the custom attributes `workAnniversary` and `employeeAge`, use the following commands.

To map the sub-attribute `employeeAge`:

```
dsconfig create-scim2-attribute-mapping \
--mapping-name urn:ietf:params:scim:schemas:extension:gluu:2.0:User.employeeAge \
--type number \
--set scim-attribute-name:employeeAge \
--set attribute-usage:create-during-realtime-sync \
--set attribute-usage:create-during-resync \
--set attribute-usage:update-during-realtime-sync \
--set attribute-usage:update-during-resync \
--set ldap-attribute-name:loginGraceLimit \
--set single-valued:true \
--set default-value:55
```

To map the sub-attribute `workAnniversary`:

```
dsconfig create-scim2-attribute-mapping \
--mapping-name urn:ietf:params:scim:schemas:extension:gluu:2.0:User.workAnniversary \
--type date-time \
--set scim-attribute-name:workAnniversary \
--set attribute-usage:create-during-realtime-sync \
--set attribute-usage:create-during-resync \
--set attribute-usage:update-during-realtime-sync \
--set attribute-usage:update-during-resync \
--set ldap-attribute-name:loginTime \
--set single-valued:true \
--set default-value:1994-12-16T10:32:00Z
```

To map the composed complex attribute:

```
dsconfig create-scim2-attribute-mapping \
--mapping-name urn:ietf:params:scim:schemas:extension:gluu:2.0:User \
--type composed-complex \
--set scim-attribute-name:urn:ietf:params:scim:schemas:extension:gluu:2.0:User \
--set attribute-usage:create-during-realtime-sync \
--set attribute-usage:create-during-resync \
--set attribute-usage:fetch \
--set attribute-usage:update-during-realtime-sync \
--set attribute-usage:update-during-resync \
--set sub-attribute-mapping:urn:ietf:params:scim:schemas:extension:gluu:2.0:User.employeeAge \
--set sub-attribute-mapping:urn:ietf:params:scim:schemas:extension:gluu:2.0:User.workAnniversary
```

After running the previous example commands, you must add the composed complex attribute to the list of attribute mappings for the appropriate SCIM 2.0 endpoint mapping.

Configure SCIM 2.0 endpoint mappings

A System for Cross-domain Identity Management (SCIM) 2.0 endpoint mapping provides information about a specific endpoint in the SCIM 2.0 server and the kinds of entries available at that endpoint.

SCIM 2.0 servers can have multiple endpoints for different kinds of entries (for example, one for users and another for groups), and if you want to be able to synchronize different kinds of entries, then you'll need a separate SCIM 2.0 endpoint mapping for each.

The following configuration properties are associated with a SCIM 2.0 endpoint mapping:

`endpoint-path`

The relative path used to access the target endpoint in the SCIM 2.0 server. This is the portion of the path that needs to be appended to the `scim-service-url` property from the SCIM 2.0 external server to get the full path to the endpoint. For example, if the full path to an endpoint to use for accessing user entries is `https://scim2.example.com/scim/v2/Users`, and if the `scim-service-url` value is `https://scim2.example.com/scim/v2`, then the appropriate endpoint-path value would be `Users`. This is required.

`schema-urn`

The URN of the SCIM 2.0 schema for the entries that are associated with this endpoint. This is required, and multiple values can be specified if there are multiple schemas associated with the endpoint.

`attribute-mapping`

The set of attribute mappings that will be used to construct the SCIM 2.0 representation of an entry from the LDAP representation of the source entry constructed by a sync class. These attribute mappings will be used when:

- Fetching an entry from the SCIM 2.0 server to determine whether the entry needs to be created or updated
- Creating a new entry
- Updating an existing entry

At least one attribute mapping must be defined, but there will probably be several.

`search-attribute-mapping`

The set of attribute mappings that will be used to construct a SCIM 2.0 filter that will be used to search for the SCIM entry that corresponds to the mapped Lightweight Directory Access Protocol (LDAP) representation of the source entry. For example, when mapping an LDAP user to a SCIM 2.0 user, you might map the `uid` LDAP attribute to the `userName` SCIM attribute, and an LDAP entry with a `uid` value of `jdoe` could result in a SCIM 2.0 search filter of `userName eq "jdoe"`. This is required, and multiple values can be provided if there should be multiple search attributes (which will be combined in an `AND` filter).

`sync-class-name`

The name of the sync class that will be used to map source entries for synchronization to the target endpoint. This is optional, and it can be omitted if the SCIM 2.0 sync destination is only associated with a single endpoint. If the SCIM 2.0 sync destination will be associated with multiple endpoints, then this property must be specified. It can be given multiple values if multiple sync classes can be used to map source entries for the same endpoint.

You can use the following example configuration change to create a SCIM 2.0 endpoint mapping:

```
dsconfig create-scim2-endpoint-mapping \
     --mapping-name "Users Endpoint" \
     --set endpoint-path:Users \
     --set schema-urn:urn:ietf:params:scim:schemas:core:2.0:User \
     --set "attribute-mapping:User Name Mapping" \
     --set "attribute-mapping:Name Mapping" \
     --set "attribute-mapping:Display Name Mapping" \
     --set "attribute-mapping:Email Address Mapping" \
     --set "attribute-mapping:Postal Address Mapping" \
     --set "attribute-mapping:Phone Number Mapping" \
     --set "search-attribute-mapping:User Name Mapping" \
```

Configure the SCIM 2.0 sync destination

The System for Cross-domain Identity Management (SCIM) 2.0 sync destination associates a SCIM 2.0 external server with a set of one or more endpoint mappings and can also specify additional configuration properties.

Available properties include:

`server`

The SCIM 2.0 external server to which changes will be synchronized. This is required.

`endpoint-mapping`

A set of one or more SCIM 2.0 endpoint mappings to use when synchronizing changes to the SCIM 2.0 server. This is required.

`query-method`

The HTTP request method that should be used when querying the SCIM 2.0 server to fetch existing entries. The value can be one of the following:

- `get` – Use the HTTP GET method to submit the query. This is the default value that will be used if the property is not specified.
- `post` – Use the HTTP POST method to submit the query.

`update-method`

The HTTP request method that should be used when applying changes to existing SCIM 2.0 entries. The value can be one of the following:

- `put` – Use the HTTP PUT method to replace the entire entry. SCIM 2.0 servers must support this method, but it is less efficient and more risky than using the PATCH method because it has greater potential of losing changes to the entry made by other SCIM 2.0 clients.
- `patch` – Use the HTTP PATCH method to specify which specific changes should be applied to the entry. This method is an optional part of the SCIM 2.0 specification, so it might not be available in all servers, but it is more efficient and safer than the PUT method, so this is the default that will be used if the property is not specified.

You can use the following example configuration change to create a SCIM 2.0 sync destination:

```
dsconfig create-sync-destination \
     --destination-name "SCIMv2 Destination" \
     --type scim2 \
     --set "server:SCIMv2 Server" \
     --set "endpoint-mapping:Users Endpoint"
```

Configure a sync pipe

A sync pipe associates a sync source, from which changes will be read, with a sync destination, to which corresponding changes will be applied.

Although there are many useful configuration properties associated with a sync pipe, including those used to control retry attempts, rate limiting, and the number of worker threads, most of those properties already have good default values. The only properties you need to specify are:

`sync-source`

The sync source from which the changes will be read. This is required.

`sync-destination`

The sync destination to which the corresponding changes will be applied. This is required.

You can use the following example configuration change to create a sync pipe:

```
dsconfig create-sync-pipe \
     --pipe-name "LDAP Source to SCIMv2 Destination" \
     --set "sync-source:LDAP Source" \
     --set "sync-destination:SCIMv2 Destination"
```

Configure sync classes

Sync classes specify how to handle different kinds of entries read from the sync source when preparing to synchronize them to the sync destination.

When synchronizing to a System for Cross-domain Identity Management (SCIM) 2.0 server, you should have at least one sync class for each endpoint. The most important configuration properties you might need to specify include:

`evaluation-order-index`

A numeric value that indicates the order in which the sync class should be evaluated relative to other classes that are associated with the same sync pipe. Each class should have a different index, and classes will be examined in ascending order from lowest index to highest. The first class that is appropriate for a given type of change (based on criteria like the base distinguished name (DN), filter, and change type) will be used. This is required, but if you only have a single sync class for a sync pipe, then you can just use the default value of 9999.

include-base-dn

An optional base DN for source entries on which this sync class can operate. For example, if you are synchronizing users from an Lightweight Directory Access Protocol (LDAP) server, and if all of the users you want to synchronize are below `ou=People,dc=example,dc=com`, then you could use that as the base DN.

include-filter

An optional filter to use to determine which kinds of entries on which this sync class can operate. If a source entry does not match this filter, the sync class will not be used. For example, if the user entries you want to synchronize all have the `inetOrgPerson` object class, then you could use a filter of `(objectClass=inetOrgPerson)`.

attribute-map

An optional attribute map to identify and convert source attributes for use in the destination entry.

> **Note**
>
> This is different from the SCIM 2.0 attribute mapping that will be used by the SCIM 2.0 sync destination in that it is more general and is not tied to any specific type of destination. In some advanced use cases, you might need to provide values for this property (especially if you need to apply transformations that SCIM 2.0 attribute mappings can't do on their own), but in many cases, the `auto-mapped-source-attribute` property will be sufficient.

auto-mapped-source-attribute

A list of the attributes that should be automatically mapped from the source entry to the destination (before any SCIM 2.0 attribute mapping is applied, which might narrow down the set of attributes that will actually be used, and which might apply additional transformations). This might be a list of specific attribute names, but you can also use the special value `-all-` to indicate that all attributes from the source entry should be mapped to destination, or the value of `-none-` to indicate that no attributes should be automatically mapped and that only those attributes referenced in the `attribute-map` property should be included. This is required.

synchronize-creates

Indicates whether to attempt to synchronize new entries created in the sync source to the destination. This property has a default value of `true`.

synchronize-modifies

Indicates whether to attempt to synchronize changes to existing entries created in the sync source to the destination. This property has a default value of `true`.

synchronize-deletes

Indicates whether to attempt to synchronize entries removed from the sync source to the destination. This property has a default value of `true`.

`attribute-comparison-method`

The method to use when comparing attributes between the source and destination versions of an entry to see if the value has changed. If specified, the value should be one of:

- `syntax-based` – Uses the syntax and matching rules for the associated attribute type to determine whether a value has changed. This is the default behavior, and it might ignore changes that aren't considered significant by the equality matching rule (for example, if the value differs only in its use of capitalization in an attribute that uses case-insensitive matching).
- `byte-for-byte` – Uses a byte-for-byte comparison of the source and destination versions of each value to determine whether it was changed. Any difference in the value will be considered significant, even if it would not have been considered significant in accordance with the syntax and matching rules.

`modifies-as-creates`

Indicates how the server should behave if an existing entry is modified in the sync source, but no corresponding version of that entry is found in the sync destination. By default, the value is `false`, and the synchronization operation will fail, leaving the entry absent from the destination. However, if this property is set to true, then the entry will be created in the destination.

`creates-as-modifies`

Indicates how the server should behave if a new entry is created in the sync source, but a corresponding version of that entry already exists in the sync destination. By default, the value is `false`, and the synchronization operation will fail, leaving the existing destination entry unchanged. However, if this property is set to `true`, then the source and destination versions of the entry will be compared, and the add might be converted into a modify if any differences are identified.

`plugin`

An optional set of plugins that can be invoked when mapping entries from the source to the destination.

You can use the following example configuration change to create a sync class:

```
dsconfig create-sync-class \
     --pipe-name "LDAP Source to SCIMv2 Destination" \
     --class-name "User Class" \
     --set include-base-dn:ou=People,dc=example,dc=com \
     --set include-filter:(objectClass=inetOrgPerson) \
     --set auto-mapped-source-attribute:-all-
```

Set the changelog startpoint for the sync source (optional)

If the sync source is an instance of the PingDirectory server, then you should tell the PingDataSync server to record the record that is currently at the end of its changelog so that real-time synchronization can begin from there after it's started.

You can do this using the `realtime-sync set-startpoint` command. For example:

```
bin/realtime-sync set-startpoint \
     --hostname sync.example.com \
     --port 636 \
     --useSSL \
     --trustStorePath config/truststore \
     --bindDN "cn=Directory Manager" \
     --bindPasswordFile directory-manager-password.txt \
     --pipe-name "LDAP Source to SCIMv2 Destination" \
     --end-of-changelog \
     --no-prompt
```

resync command">

Perform an initial bulk synchronization with the `resync` command

You can use the `resync` command to perform an initial synchronization from the source to the destination. It iterates through all entries in the source server and compares them with their counterparts in the destination server.

Any source entries that are not found in the destination are created. For entries that exist in both the source and the destination, any differences identified between them will cause the destination entry to be updated to match the source.

You can run the `resync --help` command to see the complete usage information for the tool, but in many cases, you only need to specify the name of the sync pipe. For example:

```
resync --pipe-name "LDAP Source to SCIM 2.0 Destination"
```

> **Note**
>
> Because the System for Cross-domain Identity Management (SCIM) 2.0 sync destination uses its own attribute mapping in addition to the mapping performed by the sync class, there might be cases in which the **`resync`** command believes the source and destination versions of an entry to be out of sync when they are actually in sync. This is especially true if the sync class is configured with an **`auto-mapped-source-attribute`** value of `-all-`.
> This can cause the **`resync`** command to report that a lot more entries are being modified than actually are being updated because the SCIM 2.0 sync destination performs its own comparison of the source and destination entries and might not attempt any update if it already sees that they are equivalent.
> This doesn't have any adverse effects, but you can update the `auto-mapped-source-attribute` property to explicitly list only the names of the source attributes that might be used to create the SCIM 2.0 representation of the entry.

Start real-time synchronization

After the initial synchronization is complete, you can tell the PingDataSync server to begin real-time synchronization for the sync pipe, at which point it will start watching the source for changes and making a corresponding set of updates to the destination server.

You can do this using the `realtime-sync start` command:

```
realtime-sync start \
     --hostname sync.example.com \
     --port 636 \
     --useSSL \
     --trustStorePath config/truststore \
     --bindDN "cn=Directory Manager" \
     --bindPasswordFile directory-manager-password.txt \
     --pipe-name "LDAP Source to SCIMv2 Destination" \
     --no-prompt
```

Managing Logging, Alerts, and Alarms

Each PingDirectory server supports extensive logging features to track all aspects of the PingDirectory topology.

This section contains the following topics:

Logs and log publishers

PingDirectory 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 because of being out of scope
- 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
- Accesses to the underlying database

Access loggers

Provide information about Lightweight Directory Access Protocol (LDAP) operations processed within the server. This log only applies to operations performed in the server. This includes:

- Configuration changes
- Searches of monitor data
- Bind operations for authenticating administrators using the command-line tools and the admin 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

PingDirectory 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 might be preferred.

Configure compression by setting the `compression-mechanism` property to have the value of `gzip` when creating a new logger. See Creating a new log publisher for details.

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 after the logger has been 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` command (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. You should create the encryption settings definition 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 can 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` command 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-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...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

PingDataSync 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 logs

Log file	Description
access	File-based Access Log that records Lightweight Directory Access Protocol (LDAP) operations processed by PingDataSync. Access log records can be used to provide information about problems during operation processing and 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` command.
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 because of 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` command. The file is placed wherever the command 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 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 distinguished name (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

PingDirectory 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 after it has been 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

PingDirectory 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). After this property has been 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

PingDirectory servers enable configuring log rotation and log retention policies.

Log retention

When any retention limit is reached, the server removes the oldest archived log before 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.

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"
```

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 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 might 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 because of 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 can 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 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.

PingDirectory 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 admin 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 several 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.

> **Important**
>
> SNMP is deprecated.

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. PingDirectory 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

PingDirectory 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.

Status section	Description
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.

PingDataSync status information

Status section	Description
Sync Topology	Displays information about the connected Sync topology and any standby PingDataSync 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. 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 distinguished name (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.

Enabling and configuring the 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.

About this task

You can configure the Monitoring Endpoint using the `dsconfig` command or the admin console.

Steps

- To create the StatsD monitoring Endpoint, use either of the following:

  Choose from:

  - To use the command-line, run `dsconfig` with the `create-monitoring-endpoint` option.

    This example configures 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
    ```

  - To use the admin console:

    1. From the admin console, click Show Advanced Configuration.
    2. In the Logging, Monitoring, and Notifications section, click Monitoring Endpoints.
    3. Click New Monitoring Endpoint.

When you configure Monitoring Endpoint include:

- The endpoint's host name
- The endpoint's port
- A toggle to use TCP or UDP
- A toggle to use SSL if you use TCP

  You can configure a StatsD Monitoring Endpoint with custom tags using the `additional-tags` property. This adds the defined tags to each metric message sent to the endpoint. Each tag should be created in a "key=value" format. Additional tags are appended to the end of the StatsD message. Here is a sample StatsD message with custom tags:

  ```
  example.metric:123|g|#tag1:value1,tag2:value2
  ```

  > **Note**
  >
  > You can send data to any number of monitoring endpoints.

Sending Metrics to Splunk with StatsD

About this task

> **Note**
>
> This topic applies only to the PingDirectoryProxy server.

Using the StatsD Endpoint type, you can send metric data to a Splunk installation. In Splunk, you can use Secure Sockets Layer (SSL) to secure ports that are open for StatsD. You can configure open UDP or TCP ports in Splunk to accept only connections from a certain hostname or IP address.

> **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 securely send UDP or TCP data to Splunk:

Steps

1. Send the data to a Splunk Universal Forwarder.
2. Request that the forwarder use SSL to communicate with the Splunk Indexer.

Monitor PingDataSync

PingDataSync exposes its monitoring information under the `cn=monitor` entry. Various tools can be used to surface the server's information including the admin console, JConsole, Lightweight Directory Access Protocol (LDAP) command-line tools, or SNMP. The following information is collected for PingDataSync.

PingDataSync 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 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 can 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 Java Virtual Machine (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.

Component	Description
Processing Time Histogram	Categorizes operation processing times into several user-defined buckets of information, including the total number of operations processed, overall average response time, and number of processing times between `0ms` and `1ms`.
System Information	Provides general information about the system and the JVM on which the server is running, including host name, operating system, JVM architecture, Java home, and Java version.
Version	Provides information about the product version, including build ID and revision number.
Work Queue	Provides information about the state of the PingDataSync 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-primary 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, PingDirectory 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. Before 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.

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 PingDirectory servers as pets today can take advantage of current DevOps and IaC principles to turn them into cattle.

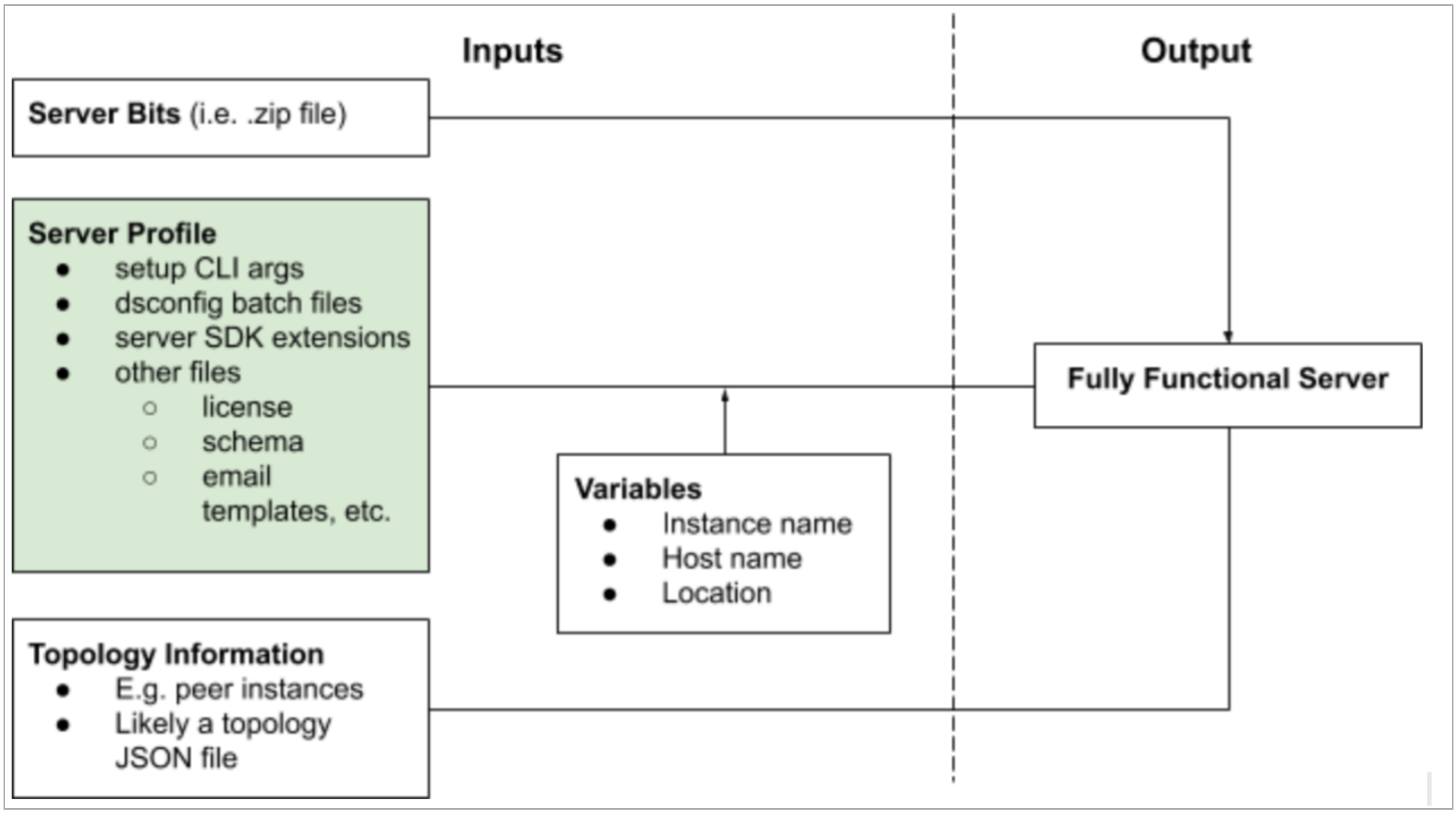
Server profiles

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 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.
- 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 before 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.

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 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/
  |-- pre-setup-dsconfig/
  |-- 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 `setup-arguments.txt` file, located in the `resources/server-profile-template.zip` archive.

When you run the `manage-profile setup` command, these arguments are passed to the server's setup tool. To view the arguments available in this file, run the server's `setup --help` command.

To provide the equivalent, non-interactive command-line interface (CLI) arguments after any prompts have been completed, run `setup` interactively. The `setup-arguments.txt` file in the server profile template contains an example set of arguments that can be changed.

`setup-arguments.txt` is the only required file in the profile.

dsconfig/ and pre-setup-dsconfig/

You can include `dsconfig` batch files in the server profile with information about changes to apply to the out-of-box system configuration. Any batch files in the server profile must include a `.dsconfig` extension. `dsconfig` batch files in the server profile can't make changes to any topology configuration, including topology admins and server instances, so you should make topology configuration changes outside of the server profile.

The batch files can be placed in the following server profile directories:

`dsconfig/`

Contains information about changes to apply after the `manage-profile` tool runs `setup`, copies any post-setup files into place, and installs any Server SDK extensions contained in the server profile. The `dsconfig/` directory should be used for most configuration changes applied to the server.

`pre-setup-dsconfig/`

Holds information about changes to apply immediately before the `manage-profile` tool runs `setup`. You should only use the `pre-setup-dsconfig/` directory when setting up the server with a pre-existing encryption settings database and applying the changes required for configuring and activating the cipher stream provider needed to access the encryption settings database.

Note

If multiple batch files are present in the directory that the `manage-profile` tool is using, the files are processed in ascending lexicographic order. For example, `00-base.dsconfig` is processed before `01-second.dsconfig`.

Tip

You can use the `config-diff` tool to create a `dsconfig` batch file that reproduces the current topology configuration.

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 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.

The `--excludeSetupArguments` option generates a server profile without a setup-arguments.txt file. This is useful when generating server profiles for use with Docker images.

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
- 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.

The `--skipValidation` option which skips the validation step when running on an offline server

> **Important**
>
> When you run the **`manage-profile replace-profile`** tool with an SDK extension included in the new profile, the tool invokes the **`setup`** command.
> The **`manage-profile replace-profile`** tool can update the server version when needed, but will fail if you attempt to downgrade the server to an earlier version. It can also 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.

Command-line tools

The server provides a full suite of command-line tools necessary for administration. 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

PingDirectory provides the following command-line tools, which you can run in interactive, non-interactive, 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`**

Command-Line Tools

`audit-data-security`	Invoke data security audit processing in order to identify potential risks or other notable security characteristics contained in directory data. This tool schedules an internal task with the server 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. The list of available auditors can be obtained using **`dsconfig list-data-security-auditors --advanced --property name`**. Either the `--includeAuditor` or the `--excludeAuditor` arguments can be used to limit the scope of the audit. Additionally, the entries scanned can be limited by specifying the backends to scan, or by specifying an LDAP filter that is used to selected entries to be processed. This tool schedules an operation to run within the PingDirectory server's process. LDAP connection options must be supplied 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.
`authrate`	Perform repeated authentications against the PingDirectory server, where each authentication consists of a search to find a user followed by a bind to verify the credentials for that user.

`backup`	Back up one or more directory server backends. Each backend backup is stored in a separate backend backup directory. A backend backup directory can contain multiple backups of the backend. Each backend backup directory contains a `backup.info` file providing information about each backup in the directory and an archive file for each backup. The name of the archive file includes both the backend ID and the backup ID. The backup ID can be provided to the backup command, or an ID is generated from a current timestamp. Each backup can be optionally compressed, encrypted, hashed or signed. A backup taken on one system can be restored on another system. This tool features both an offline mode of operation as well as the ability to schedule an operation to run within the directory 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.
`base64`	Encode raw data using the base64 algorithm or decode base64-encoded data back to its raw representation.
`check-replication-domains`	Scan the `replicationChanges` database for all known replication domains and identify any obsolete replicas still listed as part of a topology. Run this tool before upgrading replicated topologies to help avoid lockdown mode. Learn more about Discovering obsolete replicas.
`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. Information collected can include configuration files, server monitor entries, portions of log files, JVM thread stack dumps, system metrics, and other data that might be helpful in diagnosing problems, understanding server performance, or otherwise assisting with support requests. Although the tool will do its best to obscure or omit sensitive data, and the entire archive can be encrypted if you desire, you might want to review the resulting support data archive to ensure verify its contents. Further, the archive will include a summary of any potential problems or concerns that are identified in the course of collecting the support data.
`compare-ldap-schemas`	Compares the schemas of two LDAP servers to identify schema elements that might be present in one but not the other, or elements that might be present in both servers but have differences between them.

`config-diff`	Compares directory server configurations and produces a `dsconfig` batch file needed to bring the source inline with the target. Its uses include comparing multiple servers for configuration differences, producing a batch file to reconfigure a server from scratch from the out-of-the-box configuration, and comparing a local server against an expected configuration. Both the source and the target configurations can be retrieved over LDAP, accessed from the local server's file system, extracted from a specific file, or retrieved from every server in a configuration server group. Also, with the exception of accessing a configuration from a specific file, the source and/or target configurations can be compared as they existed at any point in the past, including the baseline, pre-installation configuration. Some configuration differences (those that will always differ between instances, like instance-name) are excluded by default to reduce the amount of spurious output, but these can be included by specifying the `--includeExpectedDifferences` command. Further differences can be excluded with the `--exclude` option. This tool attempts to generate a batch file that can be applied to the source server without any errors. However, there are some edge case configurations that the tool is not sophisticated enough to handle. For example, it cannot handle two peer configuration objects that would require swapping values for a property (for example, evaluation-order-index) that must be unique within the server. It will still generate a `dsconfig` batch file that includes these changes, but they might be rejected by the server. In these rare cases, the batch file can be hand edited so that it can be applied to a running server or it can be applied with the server shut down using `dsconfig --offline`.
`create-rc-script`	Create a Run Control (RC) script that you can use to start, stop, and restart the PingDirectory server on UNIX-based systems.
`create-systemd-script`	Create a systemd script to start and stop the PingDirectory server on Linux-based systems.
`dbtest`	Inspect the contents of the PingDirectory 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. Each local DB backend has a root container, identified by the backend ID. Each root container has an entry container for each base distinguished name (DN) in the backend. Each entry container has several database containers that store entries, attribute indexes and system indexes. The `dbtest` command allows the contents of root containers, entry containers and database containers to be inspected.
`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 PingDirectory server configuration. This utility offers three primary modes of operation, the interactive mode, the non-interactive mode, and the batch mode: • The interactive mode supports viewing and editing the configuration using an intuitive, menu driven environment. Running `dsconfig` in interactive provides a user-friendly, menu-driven interface for accessing and configuring the server. To start `dsconfig` in interactive command-line mode, simply invoke the `dsconfig` shell script or batch file without any arguments. • The non-interactive mode provides a simple way to make arbitrary changes to the PingDirectory server by invoking it on the command-line. If you want to use administrative scripts to automate the configuration process, then run the `dsconfig` command in non-interactive mode. • The batch mode reads multiple `dsconfig` invocations from a file and executes them sequentially. The batch file provides advantages over standard scripting in that it minimizes LDAP connections and JVM invocations that normally occur with each `dsconfig` call. You can view the `logs/config-audit.log` file to review the configuration changes made to the PingDirectory server and to use them in the batch file.
`dsjavaproperties`	Configure the JVM options used to run the PingDirectory server and its associated tools. The options managed by this tool are stored in `config/java.properties`. Typically, you should not edit that file directly. Instead, run the tool specifying `--jvmTuningParameter` arguments to customize JVM options appropriate for this system. Note that the changes will only apply to this PingDirectory server installation. No modifications will be made to your environment variables. Memory and other settings for the JVM tools, including the `start-server` tool, can be tuned during initialization by specifying one or more instances of the `--jvmTuningParameter` option when invoking this tool. Supported values are as follows: • `NONE`: Explicitly specify no parameters. • `AGGRESSIVE`: This system is dedicated to running only this server. The amount of memory allocated to this server will be computed accordingly. • `SEMI_AGGRESSIVE`: This system is shared by multiple server processes. The amount of memory allocated to this server will be computed accordingly. If no parameters are specified, the parameters specified by the previous invocation of this tool or setup will be used. Use the `NONE` option to explicitly specify no parameters.

`dsreplication`	Manage data replication between two or more PingDirectory server instances. For replication to work, you must first to enable replication using the **`enable`** subcommand. Then, you initialize the contents of one of the servers with the contents of the other using the **`initialize`** subcommand.
`dump-dns`	Obtain a listing of all of the DNs for all entries below a specified base DN in the PingDirectory 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, LDAP Data Interchange Format (LDIF) exports, and log files generated by the server.
`encryption-settings`	Manage the server encryption settings database. More information about the cipher algorithms and transformations available for use can be found in the Java Cryptography Architecture Reference Guide, as well as the Standard Algorithm Name Documentation for your chosen JDK implementation used by this server.
`enter-lockdown-mode`	Request that the PingDirectory server enter lockdown mode, during which it only processes operations requested by users holding the `lockdown-mode` privilege. While in lockdown mode, the PingDirectory server rejects all requests from users that do not hold the `lockdown-mode` privilege.
`export-ldif`	Export data from the PingDirectory server backend in LDIF form. To export data from a remote PingDirectory server, the server must be running and connection parameters must be supplied. You can specify options to include or exclude specific attributes and branches of the tree, and to include or exclude entries matching a given filter. The data can be appended to an existing file instead of overwriting it, and the output can be optionally compressed. This tool features both an offline mode of operation as well as the ability to schedule an operation to run within the PingDirectory 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.

`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. Include the following arguments to configure the output of the tool: **`--filter`** The export only includes entries matching the specified filter. **`--exportNonReversiblePasswords`** The export includes entries with non-reversible passwords. **`--exportEntriesWithoutPasswords`** The export includes entries that don't have passwords. **`--includeAdditionalAttribute`** The exported entries include the specified attribute(s) in addition to the user's DN and password. You can use `"*"` to include all user attributes and `"+"` to include all operational attributes. **`--includeVirtualAttributes`** The exported entries include the values of virtual attributes in addition to real attributes. This tool can only be used over a secure connection and when authenticated as a user with the `permit-export-reversible-passwords` privilege. The server encrypts the output using a key generated from either a user-supplied passphrase or an encryption settings definition.

`extract-data-recovery-log-changes`	Extracts changes matching a given set of criteria from a PingDirectory 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). This tool is designed to be used in conjunction with the server's data recovery log files (in the `logs/data-recovery` directory). It can be used in conjunction with other audit log files, but for best results, the logger should be configured to operate in reversible form, to include the requester DN and IP address, and to include information about any intermediate client control that might have been provided in the request. This tool must not be used with a log file that the server can update while the tool is running, or that can have some content stored in an unwritten buffer (which is especially likely if the log is compressed or encrypted). To use this tool with the server online, you should only specify a log file that has already been rotated to ensure that no more writes will be made to that file. If necessary, use the `rotate-log` tool to force the current active file to be rotated. To use this tool to revert an inappropriate set of changes, run it with `--direction revert` and an additional set of arguments that identify which changes should be reverted (for example, based on the address of the client, the authorization DN of the requester, the time frame in which the changes were applied, etc.). To use this tool to replay changes that were previously applied (for example, after restoring an old backup or importing an old LDIF file), run it with `--direction replay` and an appropriate set of arguments to select the desired set of changes. Also, make sure to use `dsreplication pre-external-initialization` before performing the restore or import and applying the changes, and then use `dsreplication post-external-initialization` after the changes have been applied. See the PingDirectory Server Administration Guide for additional information. This tool will extract changes from the selected log file (and any previously rotated files, unless the `--doNotFollowRotationChain` argument is provided) and output them in LDIF change format. If the `--outputFile` argument is provided, then the changes will be written to that file; otherwise, they will be written to standard output. If changes are to be written to a file, then the output will be compressed if the input files were compressed (unless the `--doNotCompressOutput` argument was provided), and the output will be encrypted if the input files were encrypted (unless the `--doNotEncryptOutput` argument was provided). You might want to first run the tool without specifying an output file so that you can verify that the selected changes are correct. Once you are certain that the appropriate changes have been selected, you can use a tool like `ldapmodify` or `parallel-update` to apply those changes to the server.
`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 which 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. That is, it 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 a PingDirectory server backend. Connection parameters are not required when importing to a local PingDirectory server that is not running. However, connection parameters are required if the server is remote, or if it is running locally and it is inconvenient to have to stop it for the import. You can use the options to include or exclude specific attributes and branches of the tree, and to include or exclude entries matching a given filter. The input file can be compressed. This tool features both an offline mode of operation as well as the ability to schedule an operation to run within the PingDirectory 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.
`indent-ldap-filter`	Parse a provided LDAP filter string and display it a multi-line form that makes it easier to understand its hierarchy and embedded components. If possible, the tool 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.** **The `ldap-diff` tool outputs the difference between data stored in two LDAP servers into an LDIF file. This file could be used with the `ldapmodify` command to bring the source directory server in sync with the target directory server. The specific entries to compare can be controlled with the `--searchFilter` option. In addition, only a subset of attributes can be compared by listing those attributes as trailing arguments of the command. Specific attributes can also be excluded by prepending a `^` character to the attribute. On Windows operating systems, excluded attributes must be quoted. For example: `"^attrToExclude"`. When retrieving entries from a PingDirectory server, the `@objectClassName` notation can be used to compare only attributes that are defined for a given objectClass.** **This command can be used on servers actively being modified, without reporting false positives caused by replication delays, by checking differing entries multiple times. By default, it will re-check each differing entry twice, pausing two seconds between checks. These settings can be configured with the `--numPasses` and `--secondsBetweenPass` options. The output is formatted so that delete operations come first, modify operations come next, and add operations come last. This gives the best chance that the resulting output file can be used to bring the source server into sync with the target server without causing any conflicts. This takes into account attribute uniqueness constraints as well as the requirement that child entries be deleted before parents and parents be added before children.** **The directory 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 PingDirectory 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` **For servers from other vendors, consult their documentation for configuring the proper privileges.** **The `ldap-diff` tool tries to make efficient use of memory, but it must store the DNs of all entries in memory. For directories that contain tens 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 `ldap-diff` can be customized by editing the `ldap-diff.java-args` parameter in the `config/java.properties` file and running the `dsjavaproperties` command.**

`ldap-result-code`	Display and query LDAP result codes. This tool can be used to list all known defined LDAP result codes, retrieve the name of the result code with a given integer value, or search for all result codes with names containing a given substring. At most, one of the `--list`, `--int-value`, and `--search` arguments can be provided. If none of them is provided, then the `--list` option will be chosen by default.
`ldapcompare`	Perform compare operations in an LDAP directory server. Compare operations can be used to efficiently determine whether a specified entry has a given value. The exit code for this tool will indicate whether processing was successful or unsuccessful, and to provide a basic indication of the reason for unsuccessful attempts. By default, it will use an exit code of zero (which corresponds to the LDAP 'success' result) if all compare operations completed with a result code of either 'compare false' or 'compare true' (integer values 5 and 6, respectively), but if the `--useCompareResultCodeAsExitCode` argument is provided, only one compare assertion is performed, and it yields an exit code of 'compare false' or 'compare true', then the numeric value for that result code will be used as the exit code. If any error occurs during processing, then the exit code will be a nonzero value that reflects the first error result that was encountered. The attribute type and assertion value to use for the compare operations will typically be provided as the first unnamed trailing argument provided on the command line. It should be formatted with the name or OID of the target attribute type followed by a single colon and the string representation of the assertion value. Alternatively, the attribute name or OID can be followed by two colons and the base64-encoded representation of the assertion value, or it can be followed by a colon and a less-than symbol to indicate that the assertion value should be read from a file (in which case the exact bytes of the file, including line breaks, will be used as the assertion value). The DNs of the entries to compare can either be provided on the command line as additional unnamed trailing arguments after the provided attribute-value assertion, or they can be read from a file whose path is provided using the `--dnFile` argument. If the attribute-value assertion is provided on the command line as an unnamed trailing argument, then the same assertion will be performed for all operations. If multiple types of assertions should be performed, then you can use the `--assertionFile` argument to specify the path to a file containing both attribute-value assertions and entry DNs.
`ldapdelete`	Delete one or more entries from an LDAP directory server. You can provide the DNs of the entries to delete using named arguments, trailing arguments, a file, or 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. Unless the password change method is explicitly specified (using the `--passwordChangeMethod` argument), this tool will attempt to automatically determine which method is the most appropriate for the target server using information provided in the server's root DSE. If the server advertises support for the password modify extended operation, then that method will be used. If it appears to be an Active Directory server, then an Active Directory-specific password-change method will be selected, using a regular LDAP modify operation to update the `unicodePwd` attribute with a specially encoded value. Otherwise, a regular LDAP modify operation will be used to update the value of a specified password attribute. The new password to be set for the user can be specified in one of several ways. It can be directly provided on the command line, read from a specified file, interactively prompted from the user, or automatically generated by this tool. If the new password is not specified using any of those methods, and if the password is to be updated using the password modify extended operation, then the new password field of the request will be left blank so that the server generates a new password for the user and includes it in the response to the client. If no new password is specified and some other password change method is selected, then the tool will exit with an error. The current password for the user can also be specified. This is optional, although some servers might require a user to provide their current password when setting a new one. If a current password is provided (whether given as a command-line argument, read from a specified file, or interactively requested from the user), and if a regular LDAP modify operation is used to change the password, then the resulting modify request will include a deletion of the current value and an addition of the new value. If no current password is provided, then the modify request will replace any existing password(s) with the new value.
`ldapsearch`	Process one or more searches in an LDAP directory server. The criteria for the search request can be specified in several different ways, including providing all of the details directly using command-line arguments, providing all of the arguments except the filter using command-line arguments and specifying a file that holds the filters to use, or specifying a file that includes a set of LDAP URLs with the base DN, scope, filter, and attributes to return.

`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. This tool works best with small LDIF files because it reads the entire contents of the source and target LDIF files into memory so they can be quickly compared. If you encounter an out-of-memory error while running the tool, you might need to increase the amount of memory available to the JVM used to invoke it. The amount of memory available to the JVM can be customized by invoking the JVM with the `-Xms` and `-Xmx` arguments, which specify the initial and maximum amounts of memory that it can use, respectively. These arguments should be immediately followed, without any intervening space, by an integer and a unit to specify the amount of memory to be used. The unit can be either `m` to indicate that the size is in megabytes, or `g` to indicate that it is in gigabytes. For example, `-Xms512m` indicates that the JVM should be given an initial heap size of 512 megabytes, while `-Xmx2g` indicates that it should be given a maximum heap size of two gigabytes. When invoking the **`ldif-diff`** tool included in the installation of a Ping Identity server product, you can edit the `config/java.properties` file to specify the arguments to use when invoking the JVM. After modifying the file, run the **`dsjavaproperties`** tool to ensure that those changes will be used for subsequent tool invocations.
`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`** command cannot read the changes to apply from standard input. All of the change records will be read into memory before processing begins, so it is important to ensure that the tool is given enough memory to hold those change records. However, it will only operate on a single source entry at a time, so the size of the source LDIF file does not significantly impact the amount of memory that the tool requires. Note that the tool will attempt to correctly handle multiple changes affecting the same entry. However, because it only operates on one entry at a time, it cannot always behave in exactly the same way as if it were applying the changes over LDAP to a server populated with the source LDIF file. For example, it is not possible to reject an attempt to delete an entry that has subordinates, so any delete will be treated as a subtree delete. Further, not all types of modify DN change records are supported. In particular, modify DN change records are not permitted if they target any entry that has been targeted by a previous change record (for example, renaming an entry that was created by a previous add change record). Finally, it cannot perform other types of validation, like ensuring that all of the necessary superior entries exist when adding a new entry, or ensuring that a modify DN will not introduce a conflict with an existing entry.
`ldifsearch`	Search one or more LDIF files to identify entries matching a given set of criteria.

`leave-lockdown-mode`	Request that the PingDirectory server leave lockdown mode and resume normal operation. While in lockdown mode, the PingDirectory server rejects all requests from users that do not hold the lockdown-mode privilege. Note that the PingDirectory server can place itself in lockdown mode under certain conditions (for example, if it detects a security problem like a malformed access control rule that might have otherwise resulted in exposure of sensitive data).
`list-backends`	List the backends and base DNs configured in the PingDirectory server.
`load-ldap-schema-file`	Loads 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. See the server's `config/MakeLDIF` directory for example template files. 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 will be 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, PKCS #12, PKCS #11, or BCFKS key store.
`manage-extension`	Install or update PingDirectory server extension bundles. An extension bundle is a package of extension(s) that utilize the Server SDK to extend the functionality of the PingDirectory server. Extension bundles are installed from a zip archive or file system directory. The server will be restarted if running to activate the extension(s).
`manage-profile`	Generate, compare, install, and replace server profiles. Server profiles define a format for the configuration of a server, including `dsconfig`, initial DIT, setup arguments, server SDK extensions, and other files. These are combined into one concrete structure. This tool provides subcommands that can be used to generate a new profile from an existing server, to set up a new server, and to replace an existing server's profile with a different profile. A template server profile file structure can be found in the resource/ directory.
`manage-tasks`	Access information about pending, running, and completed tasks scheduled in the PingDirectory server.

`manage-topology`	Manage the topology registry. The topology registry is a branch of the configuration DIT (`cn=Topology,cn=config`). It stores all metadata about server instances, including their instance and listener certificates, secret keys, server groups and administrative user accounts. In addition, it also stores information about the replication topology (replication server ID and replication domain ID) when replication is enabled among servers in a Directory topology. Last but not least, it stores the license key required to install the server. Changes to the topology registry on one server are automatically mirrored to other servers in the topology. The **`dsconfig`** tool, configuration API, or the management console can be used to make changes to the topology registry. This tool allows some additional capability such as exporting the contents of the registry as a JSON file.
`migrate-ldap-schema`	Migrate schema information from an existing LDAP server into a PingDirectory server instance. This tool can be used to migrate schema information from an existing LDAP server into this PingDirectory server instance. The source server can be any standards-compliant LDAPv3 server. All attribute type and object class definitions, which are contained in the source LDAP server but not in the target PingDirectory server instance, will be either added to the target instance or written to a schema file.
`migrate-sun-ds-config`	Update an instance of the PingDirectory server to match the configuration of an existing Sun Java System Directory Server 5.x, 6.x, or 7.x. This tool can be used to compare the configuration of Sun Java System Directory Server 5.x, 6.x, or 7.x and PingDirectory server instances in order to identify any differences and update the PingDirectory server configuration to more closely match that of the Sun server instance.
`modrate`	Perform repeated modifications against an LDAP directory server.
`move-subtree`	Move all entries in a specified subtree from one server to another.
`oid-lookup`	Search the OID registry to retrieve information about items that match a given OID or name. The string to use to search the OID registry should be provided as an unnamed trailing argument. All items in the OID registry will be examined, and any items that contain the provided string in its OID, name, type, origin, or URL will be matched. If no search string is provided, the entire OID registry will be displayed.

`parallel-update`	Use multiple concurrent threads to apply a set of add, delete, modify, and modify DN operations read from an LDIF file. As with other tools like **`ldapmodify`**, changes in the LDIF file to be processed should be ordered such that if there are any dependencies between changes (for example, if one add change record creates a parent entry and another add change record creates a child of that parent), prerequisite changes come before the changes that depend on them. When this tool is preparing to process a change, it will determine whether the new change depends on any other changes that are currently in progress, and if so, will delay processing that change until its dependencies have been satisfied. If a change does not depend on any other changes that are currently being processed, then it can be processed in parallel with those changes. This tool will keep track of any changes that fail in a way that indicates they succeed if re-tried later (for example, an attempt to add an entry that fails because its parent does not exist, but its parent can be created later in the set of LDIF changes), and can optionally re-try those changes after processing is complete. Any changes that are not retried, as well as changes that still fail after the retry attempts, will be written to a rejects file with information about the reason for the failure so that an administrator can take any necessary further action upon them.
`populate-composed-attribute-values`	Populate entries in one or more backends with attribute values generated by one or more composed attribute plugins. This tool uses the configuration from a specified set of composed attribute plugin instances to identify which entries to update and what changes to apply. It can be used as an alternative to exporting the data to LDIF and re-importing to ensure that existing entries have an appropriate set of composed attribute values. This tool schedules an operation to run within the PingDirectory server's process. LDAP connection options must be supplied 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.
`profile-viewer`	View information in data files captured by the PingDirectory server profiler. Profiler data files are generated by the Profiler plugin. To create these data files, set the profile-action attribute of the Profiler configuration object to 'start' to begin collection. Set the `profile-action` attribute to `stop` to end collection and have the plugin write the file to `logs/profile`.{timestamp}.

`re-encode-entries`	Re-encode all or a specified portion of the entries in a local DB backend. This tool can be used to initiate a task that will cause a local DB backend to re-encode all or a specified subset of the entries that it contains. The contents of the entries will not be altered, but this provides a useful mechanism for applying significant changes to the way that entries are actually stored in the backend (for example, to apply encoding changes if a feature like data encryption or uncached attributes or entries is enabled). This tool schedules an operation to run within the PingDirectory server's process. LDAP connection options must be supplied 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.
`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. An index must be rebuilt if the database already contains data when the index is configured. The backend containing the provided base DN must be a local DB backend. The types of indexes that can be rebuilt include attribute indexes, VLV indexes, and JSON field indexes. **Note** PingDirectory does not support the rebuilding of system indexes, regardless of whether the rebuild is attempted online or offline. If you need to rebuild a system index, you must export the backend to LDIF and re-import it. This tool features both an offline mode of operation as well as the ability to schedule an operation to run within the PingDirectory 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.
`register-yubikey-otp-device`	Registers a YubiKey OTP device with the PingDirectory 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. This tool schedules an operation to run within the PingDirectory server's process. LDAP connection options must be supplied 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.
`remove-attribute-type-from-schema`	Safely remove an attribute type definition from the server schema. The tool will perform an appropriate set of validation before actually removing the attribute type from the schema. The below conditions must be satisfied before the attribute type can be removed. • The requester must have the update-schema privilege. • The attribute type must not be referenced by any other schema element. • The attribute type must not be defined in any schema file that is included with the PingDirectory server. Only custom attribute types can be removed from the schema. • The attribute type must not be referenced in the server configuration (for example, it must not be indexed by any backend). • The attribute type must not be present in any entry that exists in any backend. This tool schedules an operation to run within the PingDirectory server's process. LDAP connection options must be supplied 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.
`remove-backup`	Safely remove a backup from the specified PingDirectory server backend. This tool deletes the specified backup archive and updates the backup descriptor accordingly. As an alternative to removing a specific backup, you can automatically remove backups outside of specified count or age criteria. The `--retainFullBackupCount` argument can be used to indicate that the specified number of full backups should be retained, and any other full backups in the directory are eligible to be removed. The `--retainFullBackupAge` argument can be used to indicate that any full backups older than the specified age are eligible to be removed.
`remove-defunct-server`	Remove a server from this server's topology. This tool will remove the specified server from the topology. In general, the `uninstall` tool should be used to remove a server from the topology. The `remove-defunct-server` tool should only be used if a prior attempt to uninstall a server was unsuccessful or the system where the server was installed is no longer available, leaving the server permanently inaccessible from the topology. If the defunct server is online and is able to reach other servers in the topology, running remove-defunct-server from it will cleanly remove it from the topology. If it cannot reach the other servers, then remove-defunct-server must also be run from one of the online servers.

`remove-object-class-from-schema`	Safely remove an object class definition from the server schema. The tool will perform an appropriate set of validation before actually removing the object class from the schema. The below conditions must be satisfied before the object class can be removed. • The requester must have the update-schema privilege. • The object class must not be referenced by any other schema element. • The object class must not be defined in any schema file that is included with the PingDirectory server. Only custom object classes can be removed from the schema. • The object class must not be referenced in the server configuration. • The object class must not be present in any entry that exists in any backend. This tool schedules an operation to run within the PingDirectory server's process. LDAP connection options must be supplied 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.
`replace-certificate`	Replace the listener certificate for this PingDirectory server instance.
`restore`	Restore a backup of a PingDirectory server backend. Only one backend can be restored at a time by the restore command. The PingDirectory server should be stopped unless task connection options are supplied for a running server. You can list the backups contained in a particular backend backup directory. A backup taken on one system can be restored on another system. This tool features both an offline mode of operation as well as the ability to schedule an operation to run within the PingDirectory 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.
`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. If the file argument is provided one or more times to specify the target log file paths, then only those log files will be rotated. If the file argument is not given, then the server will trigger rotation for all supported log files. You must have the `config-read` and `config-write` privileges to run this tool, and you must have the necessary access control rights to create and monitor entries in the task backend. This tool schedules an operation to run within the PingDirectory server's process. LDAP connection options must be supplied 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.
`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, several 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 • 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 the **`search-logs`** 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 PingDirectory server process.
`set-delegated-admin-aci`	Request that the PingDirectory server assign appropriate ACI for configured delegated administrators of the Delegated Admin API.
`setup`	Perform the initial setup for a server instance. This tool features both interactive and non-interactive modes for accepting the product license terms and initially configuring a server instance.
`start-server`	Start the PingDirectory server.
`status`	Display basic server information. This tool prints information about the server, such as version, connection handlers, and data sources. Some information might not be available if the server is not running, or if authentication credentials are missing or do not have sufficient privileges, or if the invoking user does not have sufficient file system access rights.
`stop-server`	Stop or restart the server. This tool is used to stop or restart the local instance of the server (by omitting LDAP connection options), or a remote server (by interacting with it over LDAP). In addition, this tool is used to schedule the server for shutdown at a later time using the server's task interface.
`subtree-accessibility`	List or update the set of subtree accessibility restrictions defined in the PingDirectory server.
`sum-file-sizes`	Calculate the sum of the sizes for a set of files. This tool is used to find the sum of the sizes of one or more files. If any of the files specified is a directory then it will be recursively processed.
`summarize-access-log`	Examine one or more access log files to display several metrics about operations processed within the server.
`sync-pipe-view`	PingDataSync only: Display the detailed configuration of a sync pipe or pipes in PingDataSync and all commands necessary to replicate a specified sync pipe. You can use the `sync-pipe-view` tool online with the PingDataSync server connection credentials or you can use the tool offline. The sync pipe information can be output in a text, CSV, JSON, or tab-delimited format and can be output to a file. The `sync-pipe-view` tool features both an interactive mode and a non-interactive mode.
`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 the PingDirectory server. This tool removes the entire server or individual server components from the file system. If this server is a member of a replication topology, you must first remove references to this server in the other servers using the `dsreplication disable` command.
`update`	Update a deployed server so its version matches the version of this package.
`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 PingDirectory server. Note that output generated by this tool will be LDIF, but each entry in the output will have exactly one ACI, so entries which 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. The backend containing the provided base DN must be a local DB backend. The types of indexes that can be verified include system indexes, attribute indexes and VLV indexes. Any errors found during verification are written to the output. The verification process is exhaustive and can take a long time.
`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

The PingDirectory 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 the PingDirectory server in multiple environments.

Each property takes the form of a name-value pair that defines predetermined values for a tool's options.

The PingDirectory server supports the following types of properties:

- Default properties that apply to all command-line utilities
- Tool-specific properties

Evaluation of command-line options and file options

You can specify options for a command-line tool on the command line, in a properties file, or both.

Options you specify on a tool's command line take priority over options in a properties file.

Consider the following scenarios.

Command-line options	Server uses ...
No command-line options	The options in the default `<server-root>/config/tools.properties` file.
Command-line options other than the `--propertiesFilePath <my-properties-file>` option	The command-line options, which take priority if the options are also in the `<server-root>/config/tools.properties` file. The file options for options that are only in the default `<server-root>/config/tools.properties` file.
Only the `--propertiesFilePath <my-properties-file>` option	The options in `<my-properties-file>`.
The `--propertiesFilePath <my-properties-file>` option and other command-line options	The command-line options, which take priority if the options are also in `<my-properties-file>`. The file options for options that are only in `<my-properties-file>`.
The `--noPropertiesFile` option and other command-line options	Only the options you specify on the command line, ignoring the default properties file.

Example

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
```

The server checks command-line options and file options to determine the options to use, as explained below.

- All options presented with the tool on the command line take precedence over any options in a properties file.

  In the following example, the command runs with the options specified on the command line (`--port` and `--baseDN`). With the `port` value both on the command line and in the properties file, the command-line value takes priority. 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=*)"
  ```

- If you specify the properties file using the `--propertiesFilePath` option and no other command-line options, the server uses only the options in the specified properties file:

```
$ bin/ldapsearch --propertiesFilePath bin/tools.properties \
  "(objectclass=*)"
```

- If you do not specify any command-line options, the server attempts to locate the default properties file in the following location:

```
<server-root>/config/tools.properties
```

By moving your `tools.properties` file from `<server-root>/bin` to `<server-root>/config`, you do not have to specify the `--propertiesFilePath` option. That change shortens the previous command to the following command:

```
$ bin/ldapsearch "(objectclass=*)"
```

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 Lightweight Directory Access Protocol (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.

dsconfig batch files">

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

The PingDirectory 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.**

Managing access control

The PingDirectory and PingDirectoryProxy servers provide 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 shouldn't see.

The access control model includes a privilege subsystem that provides even greater flexibility and protection.

This section 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) to determine what a user or a group of users can do with a set of entries, down to the attribute level.

The ACIs are stored in the `aci` operational attribute. 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 specify four items:

Resources

Resources are the targeted items or objects that specifies the set of entries and 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 ACI. Typically, you have multiple ACIs for a given branch of your DIT. The access control name helps describe its purpose. For example, you can configure an ACI labeled "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 directory server provides important access control features that provide added security for its entries.

Improved validation and security

The server provides an access control model with strong validation to help ensure that invalid access control instructions (ACIs) are not allowed into the server.

The server ensures that all access control rules (ACRs) added over LDAP are valid and can be fully parsed. The server rejects any operation that attempts to store one or more invalid ACIs. It also validates ACIs contained in data imported from an LDIF file. The server rejects any entry containing a malformed `aci` value.

As an additional level of security, the server examines and validates all ACIs stored in the data whenever a backend is brought online. If the server finds any malformed ACIs in the backend, it 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 because of an ACI 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 apply to entries anywhere in the server or scoped to only apply to a specific set of entries.

Global ACIs work in conjunction with ACRs stored in user data and provide a convenient way to define ACIs that span disparate portions of the directory information tree (DIT).

In the Server, global ACIs are defined within the server configuration, in the `global-aci` property of the configuration object for the access control handler. To view and manage global ACIs, use configuration tools like **`dsconfig`** and the admin console.

The global ACIs available by default in the Server include:

- Allow anyone, including unauthenticated users, to access key attributes of the root DSA-specific entry (DSE), including:
  - `namingContexts`
  - `subschemaSubentry`
  - `supportedAuthPasswordSchemes`
  - `supportedControl`
  - `supportedExtension`
  - `supportedFeatures`
  - `supportedLDAPVersion`
  - `supportedSASLMechanisms`
  - `vendorName`
  - `vendorVersion`
- Allow anyone, including unauthenticated users, to access key attributes of the subschema subentry, including:
  - `attributeTypes`
  - `dITContentRules`
  - `dITStructureRules`
  - `ldapSyntaxes`

- `matchingRules`
- `matchingRuleUse`
- `nameForms`
- `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
  - Virtual attributes only
- Allow anyone, including unauthenticated users, to request the following extended operations:
  - Get symmetric key
  - Password modify request
  - Password policy state
  - StartTLS
  - Who Am I?

Access controls for public or private backends

Depending on their intended purpose, the PingDirectory server classifies backends as either public or private.

A backend is private if one of the following applies:

- Its content is generated by the server itself, such as the root DSE, monitor, and backup backends.
- It is used in the operation of the server, such as the configuration, schema, task, and trust store backends.
- Its content is maintained by the server, such as the LDAP changelog backend.

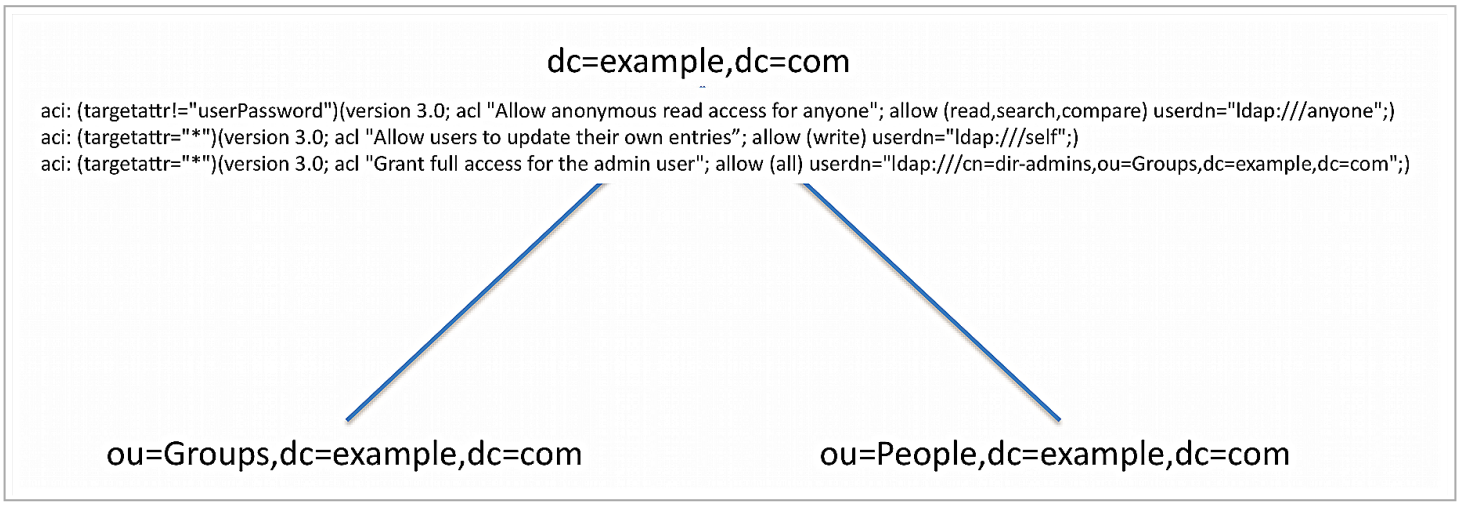
A public backend is intended to hold user-defined content, such as user accounts, groups, application data, and device data.

The 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, you should define any ACI intended to permit or restrict access to information in private backends 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 permissible operations. When a client application, such as `ldapsearch`, enters a request, the 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 has an ACI at an entry can also have other access rights available if ACIs are defined higher in the DIT and are applicable to the user. In most environments, ACIs are defined at the root of a main branch or a subtree, and not on individual entries unless absolutely required.



ACI

An access control rule has the following basic syntax:

```
aci : (targets) (version 3.0; acl "name";
        permissions
        bind rules
        ;)
```

Access Control Components

Access Control Component	Description
targets	Specifies the set of entries and attributes to which an access control rule applies. Use the following 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. Use the following 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 requester. Use the following syntax: `bind rule keyword = ||!= expression;` **Note** The bind rule syntax requires that it be terminated with a `;`.

Summary of access control keywords

This section provides an overview of the supported keywords in the server access control implementation.

Targets

A target expression specifies the set of entries and attributes to which an access control rule applies.

A target expression has three components:

```
(keyword[=||!=]expression)
```

Keyword

The keyword specifies the type of target element.

Expression

The expression specifies the items that are targeted by the access control rule.

Operator

The operator is either equal, `=`, or not-equal, `!=`.

> **Note**
> You cannot use the `!=` operator with the `targattrfilters` and `targetscope` keywords.

You can use the following keywords 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 Keyword	Description	Wildcards

`requestcriteria`	Determines whether an access control rule applies to an operation based on whether that operation matches a given request criteria definition.
	If present in an access control rule, the operator must be either "=" or "!=". The value must be enclosed in quotation marks and it must be the name or full DN of the configuration object that defines the desired request criteria.
	For example, let's say that you want to allow members of the `cn=Sales Administrators,ou=Groups,dc=example,dc=com` group to be able to read from and write to the entries for the users that are members of the `cn=Sales Employees,ou=Groups,dc=example,dc=com` group. To do this, you must first create a request criteria object that will match entries for users in the `cn=Sales Employees,ou=Groups,dc=example,dc=com` group. You can do this with the following configuration change:
	`dsconfig create-request-criteria \ --criteria-name "Requests Targeting Sales Employees" \ --type simple \ --set "any-included-target-entry-group-dn:cn=Sales Employees,ou=Groups,dc=example,dc=com"`
	With that request criteria defined, you can use a modification like the following to create the corresponding access control rule:
	`dn: dc=example,dc=com changetype: modify add: aci aci: (targetattr="*")(requestcriteria="Requests Targeting Sales Employees")(version 3.0; acl "Allow sales administrators to manage sales employees"; allow (read,search,compare,write) groupdn="ldap:///cn=Sales Administrators,ou=Groups,dc=example,dc=com";)`	

`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 can 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 can be used to indicate the set of entries to which the access control should apply.	Yes

Target Keyword	Description	Wildcards
`targetscope`	Specifies the scope of entries, relative to the defined target entries or the entry containing the ACI if there is no target, to which the access control rule should apply.	No

> **Tip**
>
> Learn more about target keywords in Working with targets.

Permissions

Permissions indicate the types of operations to which an access control rule could 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 can grant read access to targeted entries using the `allow (read)` permission. You can also deny access to the target entries and attributes using the `deny (read)` permission. You can list multiple permissions as required in the ACI.

```
allow(permission1...,permission2,...permissionN)
```

```
deny(permission1...,permission2,...permissionN)
```

You can use the following keywords in the permissions portion of ACIs:

Keyword	Description
`add`	Indicates that the access control applies to `add` operations. Learn more about Changing the allow add ACI behavior for entries.
`compare`	Indicates that the access control applies to `compare` operations and to search operations with a base-level scope that targets a single entry.
`delete`	Indicates that the access control applies to `delete` operations.
`export`	Indicates that the access control applies only to `modify DN` operations in which an entry is moved below a different parent by specifying a new superior distinguished name (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 alter the relative distinguished name (RDN) of an entry but keeps it below the same parent, such as renaming 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.

Keyword	Description
`proxy`	Indicates that the access control rule applies to operations that attempt to use an alternate authorization identity, such as 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 Simple Authentication and Security Layer (SASL) mechanism that allows an alternate authorization identify to be specified.
`read`	Indicates that the access control rule applies to search result entries returned by the server.
`search`	Indicates that the access control rule applies to `search` operations with a non-base scope.
`selfwrite`	Indicates that the access control rule applies to operations in which a user attempts to add or remove their own DN to the values for an attribute, such as users adding or removing themselves from groups.
`write`	Indicates that the access control rule applies to `modify` and `modify DN` operations.
`all`	An aggregate permission that includes all other permissions except `import`, `export`, and `proxy`. This is equivalent to providing a permission of `add`, `compare`, `delete`, `read`, `search`, `selfwrite`, and `write`.

Bind rules

The bind rules indicate whether an access control rule should apply to a given requester.

The syntax for the target keyword has three components:

Keyword

The keyword specifies the type of target element.

Expression

The expression specifies the items that are targeted by the access control rule.

Operator

The operator is either equal, `=`, or not-equal, `!=`.

You must use the semicolon delimiter symbol, `;`, after the end of the final bind rule.

```
keyword[=||!=]expression;
```

For added access control precision, you can combine multiple bind rules using the Boolean operations `AND`, `OR`, and `NOT`. The standard Boolean rules for evaluation apply:

- Innermost to outer parentheses first
- Left to right expressions

- `NOT` before `AND` or `OR`

For example, an access control instruction (ACI) with 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 users that are `uid=admin,dc=example,dc=com` and authenticate using Simple Authentication and Security Layer (SASL) EXTERNAL or access the server from a loopback interface.

```
(userdn="ldap:///uid=admin,dc=example,dc=com and (authmethod="SSL" or ip="127.0.0.1"));
```

You can use the following keywords in the bind rule portion of ACIs:

Bind Rule Keyword	Description

`authmethod`	Indicates that the requester's authentication method is taken into account when determining whether the access control rule should apply to an operation. You cannot use wildcards in this expression.
	Use the syntax `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, using 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

`connectioncriteria`	Allows or denies an operation based on whether the client connection matches a given connection criteria definition.
	If present in an access control rule, the operator must be either "`=`" or "`!=`". The value must be enclosed in quotation marks, and it must be the name or full DN of the configuration object that defines the desired connection criteria.
	For example, you can use a modification like the following to allow any client matching the "Root Users and Topology Administrators" connection criteria to have full read and write access to entries below `dc=example,dc=com`:
	`dn: dc=example,dc=com`
	`changetype: modify`
	`add: aci`
	`aci: (targetattr="*")(version 3.0; acl "Full read and write access for administrators"; allow (read,search,compare,write) connectioncriteria="Root Users and Topology Administrators";)`

`dayofweek`	Indicates that the day of the week is taken into account when determining whether the access control rule should apply to an operation. You cannot use wildcards in this expression.
	You can separate multiple day of week values by commas. Use the following syntax.
	`dayofweek = day1, day2, ...`
	day is one of the following:
	• `sun`
	• `mon`
	• `tues`
	• `wed`
	• `thu`
	• `fri`
	• `sat`
	The following example allows users who authenticate on weekdays with an SSL/TLS certificate, such as SASL EXTERNAL, 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 is taken into account when determining whether the access control rule should apply to an operation. You can use wildcards in this expression.
	You can separate multiple DNS patterns by commas. Use the following syntax.
	`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 is taken into account when determining whether the access control rule should apply to any operation. You cannot use wildcards 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 is taken into account when determining whether the access control rule should apply to an operation. You can use wildcards in this expression.
	You can separate multiple IP address patterns by commas. Use the following syntax.
	`ip [ = || != ] <ipAddressList>`
	ipAddressList is one of the following:
	• 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 is taken into account when determining whether the access control rule applies to an operation.
	Use the following syntax.
	`oauthscope [ = || != ] "<scopeIdentifier>"`
	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 grants 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";)`

`secure`	Allows or denies an operation based on whether the client is communicating with the server over a secure connection; for example, using LDAPS or StartTLS over LDAP.
	If present in an access control rule, the operator must be either "`=`" or "`!=`", and the value must be either `"true"` or `"false"` including the quotation marks. With this in mind, the `secure` bind rule takes the following forms:
	• `secure="true"` indicates that the ACI will apply only if the connection was received over a secure connection.
	• `secure="false"` indicates that the ACI will only apply if the connection was received over a non-secure connection.
	• `secure!="true"` indicates that the ACI will apply only if the connection was received over a non-secure connection.
	• `secure!="false"` indicates that the ACI will apply only if the connection was received over a secure connection.
	For example, you can use a modification like the following to allow users below `dc=example,dc=com` to update their own passwords over a secure connection:
	`dn: dc=example,dc=com`
	`changetype: modify`
	`add: aci`
	`aci: (targetattr="userPassword")(version 3.0; acl "Allow users to update`
	`their own passwords over a secure connection"; allow (write)`
	`userdn="ldap:///self" and secure="true";)`

Bind Rule Keyword	Description

`timeofday`	Indicates that the time of day is taken into account when determining whether the access control rule should apply to an operation. You cannot use wildcards in this expression.
	Use the following syntax.
	`timeofday [ = || != || >= || > || <= || < ]  <time>`
	time is one of the following:
	• 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)
	• You cannot use wildcards in this expression
	The following example allows users to update their own entries if the request is received before 12 noon.
	`aci: (targetattr="*")`
	`(version 3.0; acl "Allow users who authenticate before noon`
	`to update their own entries";`
	`allow (write) (timeofday<1200 and userdn="ldap:///self"`
	`and authmethod="simple");)`

`userattr`

Indicates that the requester's relation to the value of the specified attribute is taken into account when determining whether the access control rule should apply to an operation.
Use the following syntax.

```
userattr =  "<attrName># [ <bindType> || <attrValue> ]"
```

Where:

- *attrName* = name of the attribute for matching
- *bindType*= `USERDN`, `GROUPDN`, `LDAPURL`
- *attrValue* = an attribute value. The *attrVALUE* of the attribute must match on both the bind entry and the target of the ACI.

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 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.

Bind Rule Keyword	Description

	`aci: (targetattr="*")`
	`(version 3.0; acl "Allow a user that matches the filter to change entries";`
	`allow (write) userattr="allowedEditorCriteria#LDAPURL";)`

`userdn`	Indicates that the user's DN is taken into account when determining whether the access control rule should apply to an operation.
	Use the following syntax.
	`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 can include wildcard characters to define patterns. A single asterisk will match any content within the associated DN component, and two consecutive asterisks can 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 Lightweight Directory Access Protocol (LDAP) entry.

Access token validators are used by the following services:

- Directory REST API
- SCIM 2
- Delegated Admin
- Consent API

You can configure the PingDirectory server to accept access tokens provided by LDAP clients using the OAUTHBEARER Simple Authentication and Security Layer (SASL) authentication method.

Access token validator processing

You can configure any number of access token validators for the server.

Each access token validator has an evaluation order index, which is an integer that determines the processing priority when multiple access token validators are configured. Lower values are processed before higher values.

> **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.

Processing steps

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 Lightweight Directory Access Protocol (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.

Access token validator types

The server works with a variety of access token validators.

This section contains the following topics:

Configuring a sample 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.

Before you begin

Before using 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.

About this task

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 the directory 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.

Steps

1. In PingFederate, create a client with the following properties:
    - Client ID: Ping Identity
    - Client authentication: Client Secret
    - Allowed grant types: Access Token Validation
2. Take note of the client secret and use the directory 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:PingDirectory \
  --set "client-secret:<client secret>"
  --set evaluation-order-index:2000
  --set "identity-mapper:User ID Identity Mapper"
```

Learn more about the configuration options for a PingFederate access token validator in the PingDirectory Configuration Reference Guide.

3. 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, 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 accepts. Customize these allow lists to reflect only the signing and encryption algorithms used by your access token issuer and no others. This 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.

Learn more about the configuration options for a JWT access token validator.

Handling signed tokens

The token issuer must cryptographically sign all access tokens that the JSON web token (JWT) access token validator handles. 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:

  Choose from:

  - Store the public key as a trusted certificate in the server's local configuration using the `trusted-certificate` property.

- Provide the issuer's JSON Web Key Set (JWKS) endpoint using the `jwks-endpoint-path` property.

> **Note**
>
> To ensure that the JWT access token validator uses updated copies of the issuer's public keys, the validator checks the configured JWKS endpoint in the following cases:
>
> - When the validator initializes
> - If the validator can't find a suitable key for verification in its current set of 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

Configure the JSON web token (JWT) access token validator to accept encrypted access tokens. You must configure the access token validator with a private and 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 the PingDirectory server and provide it to your token issuer.

   Choose from:

   - To copy the public key to a file, run `dsconfig`.
   - Copy the value of the key pair's `certificate-chain` property in the admin console.

   > **Note**
   >
   > Without this public encryption key, the issuer cannot encrypt tokens that can be decrypted by the JWT access token validator.

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 similar, but you 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 the PingDirectory 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 for development or testing purposes.

A mock access token validator accepts arbitrary tokens without validating whether a trusted source issued them. This allows you 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 shows a mock access token in an HTTP request.

```
curl -k -X GET https://localhost:1443/directory/v1/Me -H 'Authorization: Bearer {"active": true, "sub":"user.1",
"scope":"email profile", "client_id":"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.

Sample 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 \
  --validator-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

# 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"
```

Learn more about the configuration options for a mock access token validator.

Third-party access token validator

To create custom access token validators, use the Server SDK. Learn more about the configuration options for a third-party access token validator.

Working with targets

The following section presents a detailed look and examples of the target access control instruction (ACI) keywords:

- `target`
- `targetattr`
- `targetfilter`
- `targattrfilters`
- `targetscope`
- `targetcontrol`
- `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, use a clause like the following.

```
(target = ldap:///uid=john.doe,ou=People,dc=example,dc=com)
```

> **Note**
>
> In most cases, you should avoid specifying a target DN. Instead, define the ACI 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, you should define it 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. 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 contain and begin with `john.d`.

  ```
  (target = ldap:///uid=john.d*,ou=People,dc=example,dc=com)
  ```

  Entries such as `john.doe,ou=People,dc=example,dc=com` and `john.davies,ou=People,dc=example,dc=com` would match the target expression.

- The following target expression matches all entries whose DN begins with `john.d` and matches the `ou` attribute.

  ```
  (target = ldap:///uid=john.d*,ou=*,dc=example,dc=com)
  ```

  Entries such as `john.doe,ou=People,dc=example,dc=com` and `john.davies,ou=asia-branch,dc=example,dc=com` would match the target expression.

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 Server:

`(targetattr="*")`

Indicates that the access control rule applies to all user attributes. Operational attributes are not automatically included in this set.

`(targetattr="+")`

Indicates that the access control rule applies to all operational attributes. User attributes are not automatically 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 does 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 Server distinguishes between these two types of attributes in its access control implementation. The PingDirectory 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.

```
(targetattr="+")
```

To include all user and all operational attributes, use both symbols.

```
(targetattr="*||+")
```

If you need to target a specific operational attribute rather than all operational attributes, include it in the values of the `targetattr` clause.

```
(targetattr="ds-rlim-size-limit")
```

If you want to target all user attributes and a specific operational attribute, use them in the `targetattr` clause.

```
(targetattr="*||ds-rlim-size-limit")
```

The following sample ACIs are placed on the `dc=example,dc=com` tree. The first ACI 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";)
```

When assigning access to user and operational attributes, it is easy to inadvertently create an access control instruction that grants far more capabilities to a user than originally intended. The following example shows the implications of the PingDirectory server not distinguishing between these attributes.

```
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 common rule and seems harmless, but has serious consequences for a PingDirectory 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 several 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 allows them to edit any other attributes in their own entry, including those excluded by rules like `uid` and `employeeNumber` in the previous example, or add, modify, or delete any entries below his or her own entry.

Because the Server does not automatically include operational attributes in the target attribute list, these types of ACIs do not present a security risk for it. 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 the `ou=engineering` attribute.

```
(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 format.

```
(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 the directory server admins group to grant the
   soft-delete-read privilege"; allow (write)
   groupdn="ldap:///cn=PingDirectory Server admins,ou=group,dc=example,dc=com";)
```

targetscope

Use the `targetscope` keyword to restrict the scope of an access control rule (ACR).

ACIs use a subtree scope by default, meaning they are applied to the target entry and all entries below it, either as defined by the target clause of the ACI or the entry in which the ACI is defined if it does not include a target. However, you can add the `targetscope` element into an ACR to restrict the set of entries to which it applies.

The following `targetscope` keyword values are allowed:

base

Indicates that the ACR applies only to the target entry and not to any of its subordinates.

onelevel

Indicates that the ACR applies only to entries that are 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 ACR applies to the target entry and all of its subordinates. This is the default behavior if no `targetscope` is specified.

subordinate

Indicates that the ACR applies to all entries below the target entry but not the target entry itself.

The following ACI targets all users to view the operational attributes present in the root DSA-specific entry (DSE):

- `supportedControl`
- `supportedExtension`

- `supportedFeatures`
- `supportedSASLMechanisms`
- `vendorName`
- `vendorVersion`

In the following example, `targetscope` is set to `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

Use the `targetcontrol` keyword to indicate whether a given request control can be used by users targeted in the ACI.

You can provide multiple OIDs by separating them with two pipe characters, optionally surrounded by spaces. When specifying control OIDs, wildcards are not allowed.

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

Use the `extop` keyword to indicate whether a given extended request operation can be used.

You can provide multiple OIDs by separating them with two pipe characters, optionally surrounded by spaces. When specifying extended request OIDs, wildcards are not allowed.

The following ACI example allows the `uid=user-mgr` to use the password modify request, `OID=1.3.6.1.4.1.4203.1.11.1`, and the StartTLS, `OID=1.3.6.1.4.1.1466.20037`, which are 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 demonstrates access controls that are commonly used in your environment.

To modify access control definitions in the server, a user must have the `modify-acl` privilege.

Administrator access

The following access control instructions (ACIs) grant members of the `cn=admins,ou=groups,dc=example,dc=com` group the following permissions:

- Add, modify, and delete entries
- Reset passwords
- 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 ACIs allow anonymous read, search, and compare on select attributes of `inetOrgPerson` entries while authenticated users can access several more. An authenticated user inherits the privileges of the anonymous ACI and can also 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";)
```

To prevent anonymous access to the directory server, set the global configuration property `reject-unauthenticated-requests` to `true`.

Delegated access to a manager

The following ACI allows 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 distinguished name (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 ACI allows 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.

```
aci: (version 3.0;acl "Application OnBehalf can proxy as another entry";
allow (proxy) userdn="ldap:///cn=OnBehalf,ou=applications,dc=example,dc=com";)
```

Note

The application user must have the `proxied-auth` privilege.

Validating ACIs before migrating data

Identify any access control instruction (ACI) syntax problems before migrating data.

Many directory servers allow for a less restrictive application of their access control instructions so that they accept invalid ACIs. For example, if an Oracle directory server encounters an access control rule that it can't parse, it ignores the rule without providing a warning, and the server might not offer the intended access protection.

The PingDirectory server rejects any ACIs that it can't interpret, which ensures that data access is limited as intended. However, this can cause problems when migrating data with existing access control rules to PingDirectory.

To validate an ACI, PingDirectory provides a `validate-acis` tool in the `bin` directory on UNIX or Linux systems or in the `bat` directory on Windows systems. The `validate-acis` tool identifies any ACI syntax problems before you migrate 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 contains a single ACI. If an entry in the input contains multiple ACIs, then it can be present multiple times in the output, each time with a different ACI value. The entries contained in the output contains only ACI values. All other attributes are ignored.

Validating ACIs from a file

Use the `validate-acis` tool to process data contained in an LDIF file.

About this task

Note

The **`validate-acis`** tool ignores all attributes except `aci` and ignores all entries that do not contain the `aci` attribute. You can use any existing LDIF file that contains access control rules.

Steps

1. Run the `validate-acis` tool and specify the input file and output file.

   Choose from:

   - UNIX or Linux: `bin/validate-acis`
   - Windows: `bat\validate-acis`

   If the output file already exists, the existing file contents are re-written. If no output file is specified, the results are written to standard output.

   Example:

   ```
   $ bin/validate-acis --ldifFile test-acis.ldif --outputFile validated-acis.ldif
   ```

   Result:

   ```
   # 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. Open the output file and review the results.

   Example:

   For example, the `validated-acis.ldif` file 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";)
   ```

   Example:

If the input file has any malformed access control instructions (ACIs), then the generated output file will show what was incorrectly entered. For example, if `userPassword` in the `test-acis.ldif` file does not have quotation marks around it, the output file reports an error. 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
```

Result:

```
# Processing complete
# Total entries examined: 1
# Entries found with ACIs: 1
# Total ACI values found: 3
# Malformed ACI values found: 1
# 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

Use the `validate-acis` tool to examine access control instructions (ACIs) in data that exists in another directory server that you are planning to migrate to the current server.

About this task

The tool helps to determine whether the PingDirectory server accepts those ACIs.

Steps

- Run the `validate-acis` tool.

  Choose from:

  - UNIX or Linux: `bin/validate-acis`
  - Windows: `bat\validate-acis`

    Provide arguments that specify:

  - The address and port of the target PingDirectory server
  - Credentials to use to bind
  - The base DN of the subtree containing the ACIs to validate

  Example:

  ```
  $ bin/validate-acis
  ```

  Result:

  ```
  # 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 Oracle to the PingDirectory server

This section describes the most important differences in access control evaluation between Oracle and the server.

Support for macro ACIs

Oracle supports macro access control instructions (ACIs), making it possible to define a single ACI that you can use to apply the same access restrictions to multiple branches in the same basic structure.

Macro ACIs are infrequently used and can cause severe performance degradation, so the server does not support them. However, you can achieve the same result by creating the same ACIs in each branch.

Support for the roleDN bind rule

Oracle roles are a proprietary, non-standard grouping mechanism that provide little value over standard grouping mechanisms.

The server does not support Oracle Directory Server Enterprise Edition (DSEE) roles and does not support the use of the `roleDN` ACI bind rule. However, you can achieve the same behavior by converting the DSEE roles to standard groups and using the `groupDN` ACI bind rule.

Targeting operational attributes

The Oracle access control model doesn't differentiate between user attributes and operational attributes.

In the Oracle access control model, using `targetattr="*"` automatically targets both user and operational attributes. Using an exclusion list like `targetattr!="userPassword"` automatically targets all operational attributes in addition to all user attributes except `userPassword`. This presents several significant security holes here users are unintentionally given access to operational attributes. In some cases, it could allow users to exempt themselves from password policy restrictions.

The server treats operational attributes differently from user attributes and never automatically includes operational attributes. For example, `targetattr="*"` targets all user attributes but no operational attributes, and `targetattr!="userPassword"` targets all user attributes except `userPassword` but no operational attributes.

You can target specific operational attributes by including the names in the list, such as `targetattr="creatorsName||modifiersName"`. You can target all operational attributes by using the `"+"` character. For example, `targetattr="+"` targets all operational attributes but no user attributes, and `targetattr="*||+"` targets all user and operational attributes.

Example

The following example searches for all immediate children of `ou=People,dc=example,dc=com`. The attributes returned are restricted to `sn`, `givenName`, and all operational attributes.

```
ldapsearch --bindDN uid=admin,dc=example,dc=com --bindPassword password \
     --baseDN ou=People,dc=example,dc=com --searchScope one '(objectclass=*)' \
     sn givenName "+"
```

Example

You can use compound filters to search for a subset of the entries in the `ou=People,dc=example,dc=com` subtree. The following example limits the returned entry amount to 200, and the server will spend no more than 5 seconds processing the request.

```
ldapsearch --bindDN uid=admin,dc=example,dc=com --bindPassword password \
     --baseDN ou=People,dc=example,dc=com --searchScope sub --sizeLimit 200 \
     --timeLimit 5 "(&(sn<=Doe)(employeeNumber<=1000))" ds-entry-unique-id \
     entryUUID
```

Returning all user and operational attributes in a schema search

Specify `"*"` in a search attribute list to represent all user attributes. Specifying `"+"` retrieves all operational attributes.

About this task

The following standards are used in PingDirectory as part of the LDAP specification.

Standard	Overview
RFC 3673	Describes the use of `"+"` to represent all operational attributes
RFC 4511	Describes the use of `"*"` to represent all user attributes

Steps

- To search the `cn=schema` entry and return all user and operational attributes, run `ldapsearch`.

  Example:

  ```
  bin/ldapsearch --baseDN cn=schema --searchScope base "(objectclass=*)" "*" "+"
  ```

Exclude attributes

The server accepts syntax, in addition to the RFC 3673, RFC 4511, and RFC 4529 standards, that allows you to exclude attributes from search results.

About this task

To exclude an attribute from the search results in PingDirectory:

Steps

- Prefix the attribute name with either `"^"` or `"!"`

  Example:

  The following example returns organizational units (OUs) that are part of the object class `group` in Colorado with the exception of OUs in Denver.

  ```
  (&(objectClass=group)(&(ou:dn:=Colorado)(!(ou:dn:=Denver))))
  ```

  Example:

  The following example returns all users that aren't `device`.

  ```
  (&(objectClass=user)(!(objectClass=device)))
  ```

  > **Note**
  >
  > To exclude all attributes associated with an object class, prefix the object class name with either `"^@"` or `"!@"`.

Specification of global ACIs

Both Oracle Directory Server Enterprise Edition (DSEE) and the server support global ACIs, which you can use 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 maintaining an identical rule in each naming context.

In DSEE, global ACIs are created by modifying the root DSA-specific entry (DSE) to add values of the `aci` attribute. In the server, you manage global ACIs with `dsconfig` referenced in the `global-aci` property of the access control handler.

Defining ACIs for non-user content

In Oracle Directory Server Enterprise Edition (DSEE), you can write to the following backends to define ACIs:

- Configuration
- Monitor
- Changelog
- Tasks

In the server, you should define access control for private backends as global ACIs, such as the following backends:

- Configuration
- Monitor
- Schema
- Changelog
- Tasks
- Encryption settings
- Backups
- Alerts

Limiting access to controls and extended operations

Oracle Directory Server Enterprise Edition (DSEE) provides limited support for restricting access to controls and extended operations.

To the extent that you can control access to controls and extended operations with access control instructions (ACIs), DSEE defines entries with a distinguished name (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 server, you can use the `targetcontrol` keyword to define ACIs that grant or deny access to controls. You can use the `extop` keyword to define ACIs that grant or deny access to extended operation requests.

Tolerance for malformed ACI values

If the server is running with less than the intended set of access control instructions (ACIs), it might prevent access to data that should be allowed or grant access to data that should be restricted. In Oracle Directory Server Enterprise Edition (DSEE), if the server encounters a malformed access control rule (ACR), it ignores the rule. This can cause the server to run with less than the intended set of ACIs. To guard against this, the server is more strict about the ACRs that it accepts.

When performing an LDIF import, the server rejects any entry containing a malformed or unsupported ACR. The server also rejects any `add` or `modify` request that attempts to create an invalid ACI.

In the unlikely event that a malformed ACI is accepted into the data, the server immediately places itself in lockdown mode. In lockdown mode, 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**
>
> To review any rejected ACIs, run the **`import-ldif`** tool with the `--rejectFile` option.

About the privilege subsystem

In Oracle Directory Server Enterprise Edition (DSEE), only the root user is exempt from access control evaluation.

While administrators can create access control instructions (ACIs) that give normal users full access to any content, they can also create ACIs that would make a portion of the data inaccessible to those users. Additionally, some tasks can only be accomplished by the root user and the capabilities assigned to the root user can't be restricted.

The server uses a privilege subsystem 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. Additionally, certain risky actions require that the requester have certain privileges in addition to the sufficient access control rights to process the operation.

These risky actions include:

- Interacting with the server configuration
- Changing another user's password
- Impersonating another user
- Shutting down and restarting the server

Identifying unsupported ACIs

To determine whether access control instructions (ACIs) are suitable for use, the server provides a **`validate-acis`** tool that you can use to examine content in an LDIF file or data in another directory server, such as an Oracle Directory Server Enterprise Edition (DSEE) instance.

When migrating data from a DSEE deployment into a server instance, use the **`validate-acis`** tool to determine whether ACIs contained in the data are acceptable. If any problems exist, update the data to redefine the ACIs so they are suitable for use in the server.

For more information, see Validating ACIs before migrating data.

Working with privileges

In addition to the access control implementation, the server includes a privilege subsystem that you can use to control what users are allowed to do.

The privilege subsystem works in conjunction with the access control subsystem to only allow privileged operations that are permitted by the access control configuration. The user must also have all of the necessary privileges.

You can use privileges to grant normal users the ability to perform certain tasks that, in most other directories, would only be allowed for the root user. The capabilities given to root users in the server are all granted through privileges, so you can create a normal user account with the ability to perform 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. To restrict root users to only the tasks that they must perform, define multiple root users in the server with different sets of privileges.

Available privileges

The following privileges are defined in the 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.
`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** 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 can still be enforced for other types of operations.

Privilege	Description
`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 must also 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 Java Management Extensions (JMX) notifications generated by the 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 server through the JMX.
`jmx-write`	This privilege is required for a user to update any information exposed by the server through the JMX. The user is also required to have the `jmx-read` privilege. **Note** 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 access control instruction (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 described in the previous description of the `ldif-export` privilege.
`lockdown-mode`	This privilege allows the associated user to request that the server enter or leave lockdown mode, or to perform operations while the server is in lockdown mode.
`modify-acl`	This privilege is required for a user to add, modify, or remove access control rules defined in the server. The server's access control configuration must also allow the user to make the corresponding change to the `aci` operational attribute.
`password-reset`	This privilege is required for one user to be allowed to change another user's password. This privilege is not required for a user to be allowed to change his or her own password. The user must also have the access control instruction privilege to write the `userPassword` attribute to the target entry.

Privilege	Description
`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 bears some security risk, because it allows users to inherit the level of access of any user, including root users and administrators. Consider restricting proxy users as described in the section Restricting proxy users.
`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 distinguished names (DNs) and all values for one or more attributes for a specified portion of the directory information tree (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.

You can assign privileges to root users in two ways:

- By default, root users can be granted a specified set of privileges.

> **Note**
>
> You can 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.

- You can grant additional privileges to individual root users and remove some automatically-granted privileges from individual root users.

The `default-root-privilege-name` property of the root distinguished name (DN) configuration object controls the set of privileges that are automatically granted to root users. By default, these privileges include:

- `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 include:

- `bypass-pw-policy`
- `bypass-read-acl`
- `jmx-read`
- `jmx-write`
- `jmx-notify`

- `permit-externally-processed-authentication`
- `permit-proxied-mschapv2-details`
- `proxied-auth`

You can change the set of default root privileges to add or remove values as necessary. This requires the `config-read`, `config-write`, and `privilege-change` privileges, and either the `bypass-acl` privilege or sufficient permission granted by the access control configuration to change the server's configuration.

Assigning additional privileges for administrators

Steps

- To allow access to the Tasks backend, set up a global access control instruction (ACI) using `dsconfig` that allows access to members of an Administrators group.

  Example:

```
$ dsconfig set-access-control-handler-prop \
  --add 'global-aci:(target="ldap:///cn=tasks")(targetattr="*||+")
        (version 3.0; 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

You can grant privileges to normal users on an individual basis.

Add the `ds-privilege-name` operational attribute to the user's entry with the names of the desired privileges. For example, the following change grants 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 must 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.

You can use the same method to grant root users privileges that aren't included in 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

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 server is a powerful feature, it might break some applications if they expect to perform an operation requiring a privilege that they do not have.

In the majority of these cases, you can assign the necessary privilege manually to the account used by that application. However, if this workaround isn't sufficient, or if you need to remove a particular privilege, such as allowing anyone to access information through Java Management Extensions (JMX) without requiring the `jmx-read` privilege, then you can disable privileges on an individual basis.

The `disabled-privilege` property in the global configuration object controls the set of disabled privileges. 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 PingDirectory server supports the Proxied Authorization Control (RFC 4370) to allow an authorized Lightweight Directory Access Protocol (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 PingDirectory server and then issue permissions that allow the LDAP client to request operations under the identity as another user. The proxied authorization control allows client applications to securely process requests without binding or re-authenticating to the server for every operation.

The PingDirectory 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. You should use the proxied authorization v2 request control based on RFC 4370.

The proxied authorization v2 control requests that the associated operation is performed as if it had been requested by another user. You can use this control in conjunction with add, delete, compare, extended, modify, modify DN, and search requests. In such case, the associated operation processes under the authority of the specified authorization identity rather than the identity associated with the client connection, such as the user as whom that connection is bound. Specify the target authorization identity for this control as an `authzid` value, either with `dn:`, followed by the distinguished name of the target user or `u:`, followed by the user name.

> **Note**
>
> Because of the security risks when using the proxied authorization control, most directory servers enforce strict restrictions on users that can request this control. If a user attempts to use the proxied authorization v2 request control without the sufficient permission, the server returns 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 (ACIs) 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. Unless your use case requires proxying root users and administrators, consider restricting proxy users as described in Restricting proxy users.

Steps

1. **Open a text editor and create a user entry that will request operations as another user. Include the `proxied-auth` privilege. Save the file as `add-user.ldif`.**

   Example:

   In this example, the user entry `uid=clientApp` will request operations as `uid=admin,dc=example,dc=com`.

   ```
   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`.**

   Example:

```
$ bin/ldapmodify --defaultAdd --filename add-user.ldif
```

3. 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. Add the file using the `ldapmodify`.

   The client application targets a specific subtree in the Directory Information Tree (DIT) for its operations. For example, a client might need access to an accounts subtree to retrieve customer information while another client might need access to another subtree, such as a subscriber subtree.

   Example:

   In this example, the client application targets the `ou=People,dc=example,dc=com` subtree.

   > **Note**
   >
   > The ACI should be on a single line of text. The example shows the ACI over multiple lines for readability.

   ```
   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.

   Prefix `dn:` to the proxying entry or `u:` to the user name.

   Example:

   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=*)"
   ```

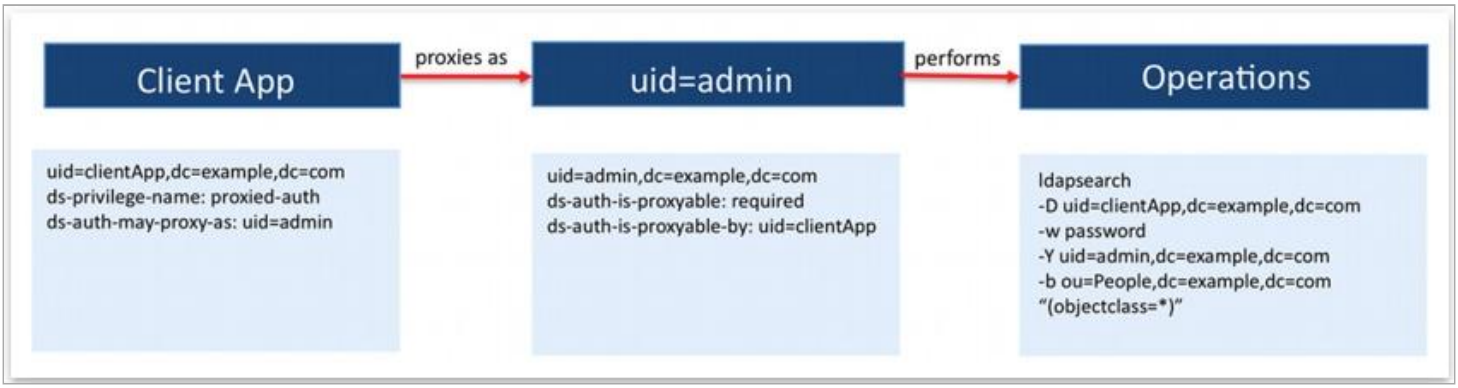
Restricting proxy users

The PingDirectory server provides a set of operational attributes that restrict the proxied authorization capabilities of a client application and its proxyable target entry.

When present in an entry, the PingDirectory server evaluates each operational attribute together to form a whitelist of potential users that can be proxied. If none of those attributes are present, the user can potentially proxy as anyone.

The PingDirectory server supports a two-tier provision system that 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 binds to the server and carries 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 through proxied authorization. If proxied authorization is allowed or required, the attributes define which client entries can proxy as the user.



Proxying operational attributes

For example, the following command demonstrates the client application `uid=clientApp` requesting to search the `ou=People,dc=example,dc=com` branch as the user `uid=admin`.

```
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 PingDirectory 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 previous figure, the `uid=clientApp` entry has a `ds-auth-may-proxy-as` attribute with a value of `uid=admin`, meaning the client app user can 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`, meaning 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`, meaning it can only be requested by the `uid=clientApp` entry. When both sets of attributes have been satisfied, the `uid=clientApp` can bind to the server as the authenticated user.

Next, the PingDirectory server performs access control instruction (ACI) evaluation on the branch and determines if the requested user has access rights to the branch. If the `uid=clientApp` entry can access the branch, the search request is processed.

About the ds-auth-may-proxy-as-* operational attributes

The PingDirectory server first evaluates the list of potential users that can be proxied for the authenticated user depending 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 distinguished names (DNs) that the associated user can proxy as. For example, you can 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 can proxy as. For example, you can 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 you can specify more than one group. 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 can proxy as. For example, 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 you can specify more than one LDAP URL.

```
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 PingDirectory server evaluates 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 fails.

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`

Operations can be proxied as this user.

`prohibited`

Operations can't be proxied as this user.

`required`

The account cannot authenticate directly but can 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.

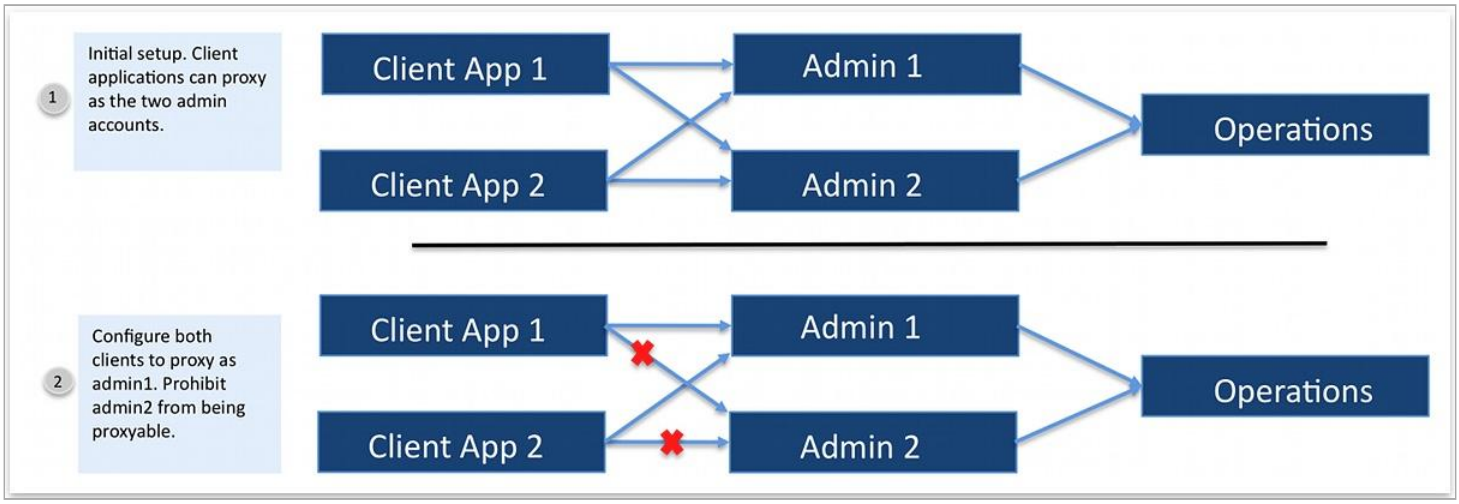
Restricting proxied authorization for specific users

Use the following example scenario with proxied users to restrict proxied authorization.

About this task

To illustrate how the proxied authorization operational attributes work, 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`. Add the `ds-auth-may-proxy-as-*` and `ds-auth-is-proxyable-*` attributes to these entries to restrict how each account can use proxied authorization.

For example, the two client applications can continue to proxy the `uid=admin1` account but the `uid=admin2` account are no longer able to be used as a proxied entry.



Proxy Users Example Scenario

Steps

1. **Set up two user entries, `uid=clientApp1` and `uid=clientApp2`, with the `proxied-auth` privilege assigned to them.**

   The user entries will proxy as the `uid=admin1` and `uid=admin2` accounts to access the `ou=People,dc=example,dc=com` subtree.

   1. **Open a text editor and create an LDIF file.**
   2. **Add the file using the `ldapmodify` tool.**

      Example:

      > **Note**
      >
      > In this example, **…** indicates that other attributes present in the entry are not included 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. **Assign the access control instruction (ACI) for each client application to the subtree, `ou=People,dc=example,dc=com`.**

   > **Note**
   >
   > ACIs should be on one line of text. The following example displays ACIs over multiple lines for readability.

   Example:

```
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.

   Example:

   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. Limit the proxied authorization capabilities for each client application.

   In the following example, the `ds-auth-may-proxy-as` attribute specifies that `uid=clientApp1` can proxy as the `uid=admin1` entry.

   1. Open a text editor and create the following LDIF file.
   2. Update the `uid=clientApp1` entry to add the `ds-auth-may-proxy-as` attribute.

      > **Note**
      >
      > `ds-auth-may-proxy-as` is multi-valued.

   3. Save the file and add it using `ldapmodify`.

      Example:

```
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, but specify the `ds-auth-may-proxy-as-url` attribute.

   The client entry can proxy as any distinguished name (DN) that matches the LDAP URL.

Example:

```
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. To illustrate the `ds-auth-proxyable-by-group` attribute, create a group of client applications that has `uid=clientApp1` and `uid=clientApp2` as its `uniquemembers`.

   Example:

   This example sets 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 can be proxied as.

   1. Open a text editor and create the following LDIF file.
   2. Use the `ds-auth-is-proxyable` to make the `uid=admin1` a required proxyable entry, meaning that it can only be accessed by some form of proxied authorization.
   3. Use the `ds-auth-is-proxyable-by` attribute to specify each DN that can proxy as `uid=admin1`.
   4. Save the LDIF file and add it using `ldapmodify`.

      Example:

```
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*)
```

Note

The example includes all three types of `ds-auth-is-proxable-by-*` attributes as an illustration, but only one type of attribute is necessary if they all target the same entries.

8. Prohibit proxying for the `uid=admin2` entry.

    1. Open a text editor and create the following LDIF file.
    2. Set the `ds-auth-is-proxyable` attribute to `prohibited`.
    3. Save the LDIF file and add it using `ldapmodify`.

    Example:

    ```
    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.

    1. To return a successful operation, run a search with `uid=clientApp1` or `uid=clientApp2` that proxies as `uid=admin1`.
    2. To return an unsuccessful operation, run a search for `uid=clientApp1` that proxies as `uid=admin2`.

    Example:

    ```
    $ 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=*)"
    ```

    Result:

The operation is unsuccessful because `uid=admin2` does not match 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`.

```
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`.

Example:

```
$ 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=*)"
```

Result:

The operation is unsuccessful because the `ds-auth-is-proxyable:prohibited` operational attribute states that `uid=admin2` is not available for proxied authorization.

```
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 PingDirectory server supports the use of parameterized access control instructions (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, consider 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";)
```

This parameterized ACI enables:

- Members of a group with the distinguished name (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 the 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 tries to read the `uid=user.1,o=acme,dc=example,dc=com` entry, this ACI is 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.

You can replace attribute values from the target DN with different variables, `($#)`, and then reference those variables in the group DN or user DN. Construct the string representation of a parameter variable as follows:

1. An open parenthesis
2. A dollar sign
3. A positive integer
4. A closing parenthesis

Consider the following 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 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 relative distinguished names (RDNs) in a parameterized target. A given RDN can have at most one parameter variable as its attribute value. A given parameter variable can 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 only applies 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 RDN of the parameterized target's DN. For example, if an ACI with a parameterized target is 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 cannot be pattern ACIs, meaning they cannot contain wildcards (`*`).
- RDNs that are parameterized must be single-valued. For example, a given parameterized RDN cannot 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 distinguished name (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 `($attr.attrName)` macro expands only the attribute's value so that you can extract values from the target DN and build relative distinguished names (RDNs) with different attribute names and types. Because the `($attr.attrName)` macro extracts only the attribute's value, you can combine it with a type other than what is in the target DN.

Managing servers and certificates

The following provides a detailed description of PingDirectory server certificate types.

The 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, such as `--bindDN` and `--bindPasswordFile`
> - Details about the key store, PEM, or DER file 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-certificate-file <path>`	The path to the PEM or DER file that contains the new certificate chain. Either this argument or `--source-key-store-file <path>` must be provided. If you are providing a certificate chain with multiple certificates, order the chain with the server certificate first. Each subsequent certificate should be the issuer for the previous certificate. You can provide `--source-certificate-file` multiple times if the certificates in the chain are in separate files rather than one file.

Argument	Description
`--source-private-key-file <path>`	The path to the PEM or DER file that contains the private key that correlates to the new PEM or DER server certificate. This argument must be provided when including `--source-certificate-file <path>`.
`--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. Either this argument or `--source-certificate-file <path>` must be provided.
`--source-key-store-password <password>`	The clear-text password needed to access the contents of the source key store.
`--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, PEM, or DER file. 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, PEM, or DER file. 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.

Argument	Description
`--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, though these providers remain separate and distinct.
`--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.
`--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 illustrates what you see when you run the `replace-certificate replace-listener-certificate` command with the `--help` argument:

```
replace-certificate replace-listener-certificate \
     --bindDN uid=admin,dc=example,dc=com \
     --bindPasswordFile admin-password.txt \
     --source-key-store-file new-listener-certificate-keystore.jks \
     --source-key-store-type JKS \
     --source-key-store-password-file new-listener-certificate-keystore.pin \
     --source-certificate-alias new-listener-cert \
     --key-manager-provider JKS \
     --trust-manager-provider JKS \
     --target-certificate-alias server-cert
```

The following example illustrates replacing a PEM-encoded listener certificate:

```
replace-certificate replace-listener-certificate \
     --bindDN cn=Directory Manager \
     --bindPasswordFile admin-password.txt \
     --source-certificate-file new-listener-certificate-chain.pem \
     --source-private-key-file new-private-key.pem \
     --source-private-key-password-file encrypted-private-key-password.txt \
     --key-manager-provider JKS \
     --trust-manager-provider JKS \
     --target-certificate-alias server-cert
```

- The following example demonstrates using the `replace-certificate` 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
bHOHEP6AAAAAAAAArOgjH+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
```

- 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.

Repairing broken listener certificate trust in replication

PingDirectory replication configuration depends on listener certificates. When certificate trust is broken between servers, replication and client traffic continues, but topology changes are disrupted.

PingDirectory uses a mirrored subtree service to keep configuration subtrees synchronized across a replicated topology. This synchronization depends on listener certificates being trusted between servers. If you discover (for example, during a topology change or from a peer connection alert) that these certificates are out of sync, the certificate trust is broken and the topology can't be properly mirrored. For example, when one server tries to send changes to trusted servers, the operation fails if the receiving servers have a different master than the sending server, or don't have a master at all.

Note

Common causes of a broken mirrored subtree:

- Replacing one or more LDAPS connection handler certificates before adding the listener certificate trust to the topology
- Updating the key store on disk without running the `replace-certificate` tool
- Replacing the listener certificate trust with the new certificate without including the old certificate
- Using an older or untested custom script to replace the certificate
- Making a typo when replacing the certificate
- Allowing the server certificate to expire

When you have a large number of servers in your environment, and your listener certificates fall out of sync, repairing trust and restoring topology mirroring can become complex. Rollback after a certificate rotation might not be possible because the new certificate is already serving client traffic. To troubleshoot, you check the logs and statuses for multiple servers. After you find the root cause, you update all server members lacking the appropriate trust, but you have to force each of these servers as master because the topology is read-only, due to the lack of trust.

Instead of following this long and complicated troubleshooting procedure, you can use the `repair-topology-listener-certificates` CLI tool. This tool helps restore trust and bring replicated environments back to a working state, greatly reducing the effort required to fix this issue.

Using the `repair-topology-listener-certificates` tool

Before you begin

Forced masters

The `repair-topology-listener-certificates` tool can fail if there are multiple masters in the topology. You must ensure no servers have `force-as-master-for-mirrored-data` set to `true` in the global configuration of any server in the topology.

Topology credentials

To update all server instances, you must supply an ID that exists on all servers in the topology and has the same password. If your root user accounts have differing passwords on all server nodes, you can use the topology administrator credentials, which consist of the same ID and password across the cluster.

About this task

You can use the `repair-topology-listener-certificates` tool to repair a broken PingDirectory topology by replacing the listener certificates for one or more server instances. The tool connects to each server in the topology and collects the certificates that they present during TLS negotiation for LDAPS, LDAP StartTLS, and HTTPS. It then examines the topology registry for each server to identify any updates that need to be made and attempts to apply those updates.

Note

Because you might need to run the tool multiple times to repair trust across the entire topology, each execution generates log files and LDIF change files to a separate directory, so that you can correlate changes to a specific run. By default, these subdirectories are stored within the current working directory and are named as follows: `repair-topology-listener-certificates-<timestamp>` .

Important arguments for `repair-topology-listener-certificates` are listed in the following tables. See the CLI for a complete list.

LDAP connection and authentication arguments

Argument	Description
`--hostname <host>`	The IP address or resolvable name to use to connect to a specific instance of the PingDirectory server. If you don't provide this value, the tool uses `localhost`.
`--port <port>`	The port to use to connect to the server. If you don't provide this value, the tool uses a default value of `389`.
`--bindDN <DN>`	The DN to use to bind to the server when performing simple authentication.
`--bindPassword <password>`	The password to use to bind to the server when performing simple authentication or a password-based SASL mechanism.
`--bindPasswordFile <path>`	The path to the file containing the password to use to bind to the server when performing simple authentication or a password-based SASL mechanism.
`--promptForBindPassword`	Indicates that the tool should interactively prompt the user for the bind password.
`--useSSL`	The tool uses SSL when communicating with the server.
`--useStartTLS`	The tool uses StartTLS when communicating with the server.
`--defaultTrust`	The tool uses the JVM's default trust store, and optionally an additional trust store specified using the `--trustStorePath` argument, to non-interactively determine whether to trust any certificate chain presented during TLS negotiation. If the chain can't be trusted based on any of those sources, then negotiation fails without prompting the user.
`--trustAll`	The tool trusts any certificate presented by the server.
`--keyStorePath <path>`	The path to the file to use as the key store for obtaining client certificates when communicating securely with the server.
`--keyStorePassword <password>`	The password to use to access the key store contents.
`--keyStorePasswordFile <path>`	The path to the file containing the password to use to access the key store contents.
`--promptForKeyStorePassword`	Indicates that the tool should interactively prompt the user for the password to use to access the key store contents.

Argument	Description
`--keyStoreFormat <format>`	The format for the key store file. For example, JKS, PKCS12, PKCS11, or BCFKS.
`--trustStorePath <path>`	The path to the file to use as trust store when determining whether to trust a certificate presented by the server.
`--trustStorePassword <password>`	The password to use to access the trust store contents.
`--trustStorePasswordFile <path>`	The path to the file containing the password to use to access the trust store contents.
`--promptForTrustStorePassword`	Indicates that the tool should interactively prompt the user for the password to use to access the trust store contents.
`--trustStoreFormat <format>`	The format for the trust store file. For example: JKS, PKCS12, PKCS11, or BCFKS.
`--verifyCertificateHostnames`	Indicates that the tool should verify that the host name or IP address used to establish connections to the server matches an address for which the server's TLS certificate was issued.
`--certNickname <nickname>`	The alias of the client certificate in the key store to present to the server for SSL client authentication.
`--enableSSLDebugging`	Enables Java's low-level support for debugging SSL/TLS communication. This is equivalent to setting the `javax.net.debug` property to `all`.
`--saslOption <name=value>`	A name-value pair that provides information for the server to use when performing SASL authentication.
`--useSASLExternal`	The tool uses the SASL EXTERNAL mechanism to authenticate.
`--helpSASL`	Provides information about the supported SASL mechanisms, including the properties available for use with each.

Usage arguments

Argument	Description
`--dryRun`	The tool identifies any updates that need to be applied to the topology registry but doesn't attempt to apply them. The identified changes are written as LDIF files in the output directory.

Argument	Description
`--forceAsMasterDelay <delay>`	The length of time that the tool should wait to update topology data for a server instance after forcing that instance to temporarily act as the topology master. If you include this argument, you must supply a value that includes an integer followed by a time unit. For no delay, supply a value of `0 seconds`. If you don't include this argument, the tool uses a default delay of 10 seconds.
`--interactive`	Launches the tool in interactive mode.
`--outputDirectory <path>`	The path to a directory where the tool should write output files, including a log file with detailed information about the processing performed by this tool and LDIF files with any changes to be applied. If you don't supply this argument, the tool creates a subdirectory named `repair-topology-listener-certificates-<timestamp>` in the current working directory.

Steps

1. (Optional) To review the changes needed before applying them, run the tool using the `--dryRun` argument.

   Example:

   ```
   $ bin/repair-topology-listener-certificates \
     --hostname ds1.example.com \
     --port 636 --useSSL \
     --trustStorePath /path/to/ds1/trust/store \
     --bindDN "cn=Directory Manager" \
     --bindPasswordFile /path/to/password.txt \
     --dryRun
   ```

2. Run the tool to apply the necessary topology registry changes.

   Choose from:

   - Remove the `--dryRun` argument and re-run the command to apply the changes identified during the dry run.
   - Build a command using the arguments needed from the previous tables and run the command to identify and apply changes.
   - Run `repair-topology-listener-certificates --interactive` to enter interactive mode and enable the CLI to prompt you for any information the tool needs to run.

3. Review the tool output and verify whether the topology trust has been restored:

   If trust is fully restored, no further steps are needed.

   Troubleshooting:

If trust hasn't been fully restored:

1. Review the command and make any adjustments needed.
2. Re-run the command.
3. Return to the beginning of this step.

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
- Has a maximum key size of 3072 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
- Certificates with an RSA key that is larger than 3072 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>`

  The following example illustrates what you see when you run `replace-certificate replace-inter-server-certificate` with the `--help` argument:

```
replace-certificate replace-inter-server-certificate \
     --bindDN uid=admin,dc=example,dc=com \
     --bindPasswordFile admin-password.txt \
     --source-key-store-file new-inter-server-certificate-keystore.jks \
     --source-key-store-type JKS \
     --source-key-store-password-file new-inter-server-certificate-keystore.pin \
     --source-certificate-alias new-inter-server-cert
```

- 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

The 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.

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.

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.

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 plain-text 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

The 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 pre-populated 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:

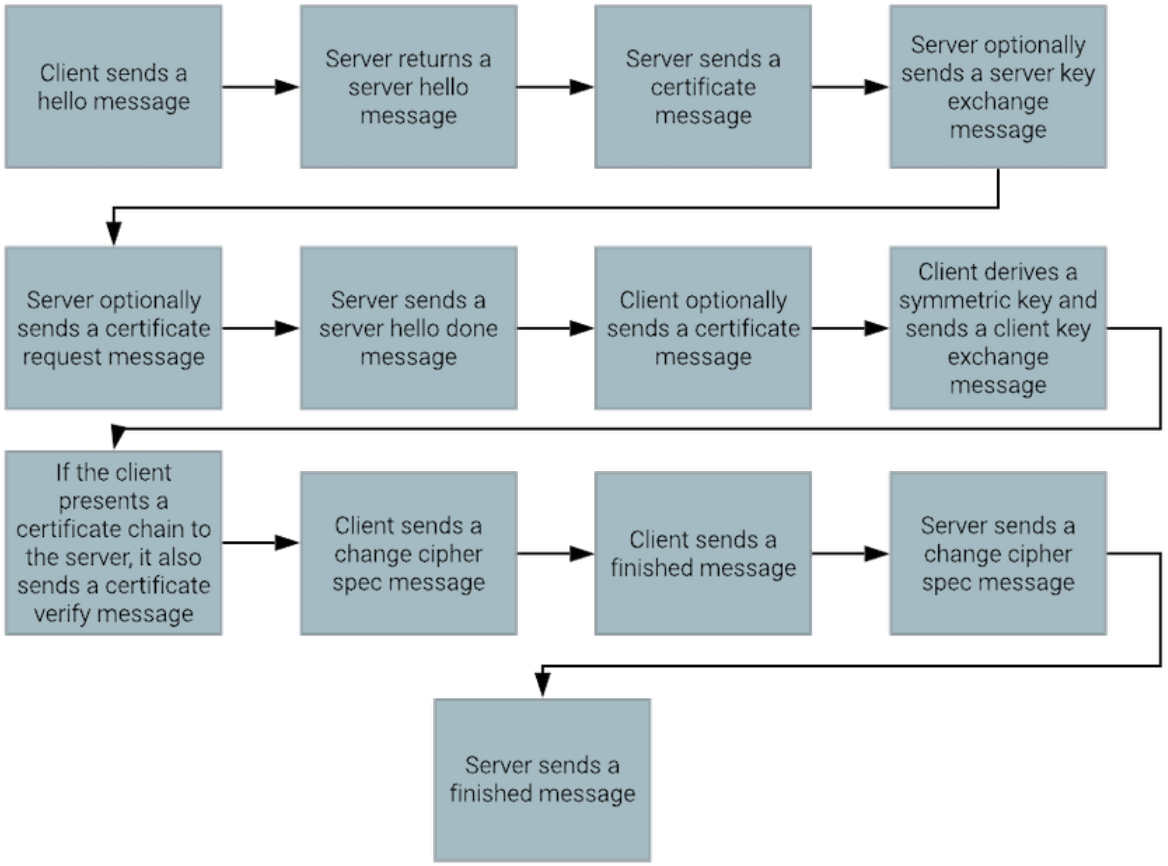
- Unless the client presents its own certificate chain to the server, the server cannot encrypt the data that it sends back to the client.
- Public key encryption is considerably slower than symmetric encryption, in which the same key is used for both encryption and decryption. Public key encryption 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, the server does not support this strategy.

The manage-certificates tool

The 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.

Subcommand	Function
`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 plain-text 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.

  Example:

  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.
```

Example:

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:
   -207531030019209390598003353674157617387647003537725397654050699787263240396 4
Elliptic Curve Y-Coordinate:
   670793565039084272923789184408894120026594857316835707373651279535545085537 3
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.
`--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.

Argument	Description
`--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.
`--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.

Argument	Description
`--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.
`--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.

Argument	Description
`--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`

Example

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 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.

   Example:

   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.

   Example:

   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.

Example:

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 admin console on the specified TCP port.

Argument	Description
`--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.
`--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 the 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.

   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. Learn more in Certificate chains.

2. Configure the key and trust manager providers.

3. Configure connection handlers.

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. In most cases, you can update the appropriate key manager provider to reference the key store that you plan to use, as shown in the following 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, as shown in the following 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.

Caching key and trust managers

When you create key and trust manager providers, caching is enabled by default, allowing the manager providers to avoid loading key store and trust store files from disk when establishing connections to process requests.

Invalidating the cache

The manager provider reloads files from the configured key store or trust store and refreshes the cache under any of the following conditions:

- The cached manager for the configured store has a `null` value.
- The path to the cached store doesn't match the path of the configured store.
- The length of the cached store doesn't match the length of the configured store.
- The last-updated time for the cached store doesn't match the last-updated time for the configured store.

Enabling or disabling caching (optional)

You can define whether caching is enabled by using the `enable-key-manager-caching` or `enable-trust-manager-caching` properties. Supply a value of `false` to disable caching, causing manager providers to load managers for each connection. Supply a value of `true` to re-enable caching.

To create a key manager provider with caching disabled, supply the `enable-key-manager-caching` property with a value of `false`, as shown in the following example:

```
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 \
    --set enable-key-manager-caching:false
```

To create a trust manager provider with caching disabled, supply the `enable-trust-manager-caching` property with a value of `false`, as shown in the following example:

```
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 enable-trust-manager-caching:false
```

To re-enable caching, set a value of `true` for the same caching property used to create the manager provider, as shown in the following example:

```
dsconfig set-trust-manager-provider-prop \
    --provider-name JKS \
    --set enable-trust-manager-caching:true
```

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.

  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 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.

  Example:

  ```
  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
  ```

  Example:

  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 the 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 the 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.
`--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.

Argument	Description
`--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.**
`--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 the PingDirectory server.

Example

```
$ 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: 20191113061149Z

# 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.

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.

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.

Using low-level TLS debugging

Use tools other than the command-line tools that are provided with the 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 the 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.

Enabling low-level debugging

About this task

> **Note**
>
> This topic applies only to the PingDirectoryProxy server.

> **Note**
>
> Because this approach requires multiple server restarts, it is not a popular option. 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.

To enable low-level debugging:

Steps

1. In the `config/java.properties` file, add `-Djavax.net.debug=all` to the `start-server.java-args` property.

2. Run `bin/dsjavaproperties`.
3. Restart the server.

Using the debug log publisher

To obtain more information for troubleshooting without a restart, use the server's built-in debugging support.

About this task

> **Note**
>
> This topic applies only to the PingDirectoryProxy server.

To use the debug log publisher:

Steps

1. To enable the debug log publisher and set the debug target, run the following configuration changes:
    1. Run `dsconfig` with the `create-debug-target` option.

        Example:

        ```
        dsconfig create-debug-target \
             --publisher-name "File-Based Debug Logger" \
             --target-name com.unboundid.directory.server.extensions.TLSConnectionSecurityProvider \
             --set debug-level:verbose
        ```

    2. Run `dsconfig` with the `set-log-publisher-prop` option.

        Example:

        ```
        dsconfig set-log-publisher-prop \
             --publisher-name "File-Based Debug Logger" \
             --set enabled:true
        ```

        Result:

        The `logs/debug` file captures a substantial amount of information about the TLS-related processing that the server is performing. Although this file doesn't provide as much detail as the Java virtual machine's (JVM) built-in debugging information, it might help to pinpoint the cause of the problem and to identify potential solutions.

2. To disable the debug log publisher and remove the debug target, run the following configuration changes:
    1. Run `dsconfig` with the `set-log-publisher-prop` option.

        Example:

        ```
        dsconfig set-log-publisher-prop \
             --publisher-name "File-Based Debug Logger" \
             --set enabled:false
        ```

2. Run `dsconfig` with the `delete-debug-target` option.

Example:

```
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 a graphical tool that runs on a variety of platforms and provides support for understanding TLS communication. Even if you can't decipher the encrypted content, you can view at least some of the handshake messages.

Note

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.

Managing SCIM 1.1 and 2.0 servlet extensions

The PingDirectory and PingDirectoryProxy servers provide 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 securely and quickly. SCIM is an alternative to LDAP, allowing identity data provisioning between cloud-based applications over HTTPS. One servlet 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.

SCIM 1.1 servlet extension configuration

This 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.

Understanding the basic concepts of SCIM can help you use the SCIM 1.1 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.

> **Important**
>
> Support for SCIM 1.1 has been deprecated and will be removed in a future release.

The 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

SCIM Feature	Supported
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 server extension to the basic SCIM functionality.

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 PingDirectory 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 (for example, 'Users' and 'Groups'), only LDAP object classes (for example, `top`, `domain`, `inetOrgPerson`, or `groupOfUniqueNames`), or both. Because SCIM and the Identity Access API have different schemas, if both are enabled, there can 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 (for example, `top --> person --> organizationalPerson --> inetOrgPerson`, and so forth), a client application can access an entry in multiple ways because of the different paths/URIs to a given resource.

This chapter provides information on configuring the SCIM and the Identity Access API services on the server.

Before you begin

To set up your system for Cross-domain Identity Management (SCIM) servlet extension, the server provides a `dsconfig` batch file at `<server-root>/config/scim-config-ds.dsconfig` for PingDirectory and at `<server-root>/config/scim-config-proxy.dsconfig` for PingDirectoryProxy.

The script runs a series of commands that:

- Enable the HTTP Connection Handler and SCIM HTTP Servlet Extension.
- Increase the level of detail logged by the HTTP Detailed Access Log Publisher.
- Add access controls to allow access to LDAP controls used by the SCIM servlet.
- Add support to the request processor for LDAP controls used by the SCIM servlet.
- Set the subordinate base distinguished name (DN) property of the root DSE so that SCIM requests can be authenticated using LDAP `uid` values.

> **Note**
>
> Edit the **`dsconfig`** batch file before running the details of your deployment.

Creating your own SCIM 1.1 application

To create your own System for Cross-domain Identity Management (SCIM) 1.1 application, you must download the SCIM 1.1 SDK.

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, and is available for download. It provides developers with a simple, concrete API to interact with a SCIM service provider.

> **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`** configuration file, but not if the SCIM attribute is a core SCIM attribute.

Configuring SCIM 1.1

This section discusses details about the PingDirectory server and PingDirectoryProxy server implementations of the SCIM protocol.

Before reading these topics, familiarize yourself with the SCIM Protocol specification, available on the Simplecloud website.

Configuring SCIM 1.1 for PingDirectory

The PingDirectory 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 (for example, 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.

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 PingDirectory 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 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.

   Example:

   ```
   $ 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.

   Example:

```
$ 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.

   Example:

   ```
   $ 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:

   Example:

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

5. Turn on the connection handler.

   Example:

   ```
   $ 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.

   Example:

```
$ 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.

   Example:

```
$ 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.

   Example:

```
$ 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.

   Example:

```
$ 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.

    Example:

    ```
    $ 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.

    Example:

    ```
    $ 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:

    Example:

    ```
    $ 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`:

    Example:

```
$ 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:

   Example:

   ```
   $ bin/dsconfig set-http-servlet-extension-prop \
       --extension-name SCIM \
       --set entity-tag-ldap-attribute:ds-entry-checksum
   ```

   Result:

   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.

   Example:

   ```
   $ bin/dsconfig --batch-file config/scim-config-ds.dsconfig
   ```

Configuring SCIM 1.1 for PingDirectoryProxy

Steps

1. To allow read access to operational attributes used by the SCIM Servlet Extension, add access controls on each of the backend servers before you enable the System for Cross-domain Identity Management (SCIM) servlet extension.

   Example:

   Instead of the `dsconfig` interactive equivalent, the following example uses the non-interactive command to add access control instructions (ACIs) .

```
$ 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. To enable the SCIM servlet extension, run the `dsconfig` batch file on the server.

   Example:

   ```
   $ bin/dsconfig --batch-file config/scim-config-proxy.dsconfig
   ```

3. Edit the `dsconfig` batch file to use the correct request processor name and base distingushed names (DNs) for the `set-request-processor-prop` and `set-root-dse-backend-prop` commands.

   Learn more in Configuring LDAP control support on all request processors and SCIM 1.1 servlet extension authentication.

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.

   Example:

   ```
   $ 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.

   Example:

   ```
   $ 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, enable new access controls required by SCIM for read access to operational attributes with the following command.

> **Note**
>
> If this access control instruction (ACI) is not present, problems will occur when a SCIM client tries to authenticate with a non-root distinguished name (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

You must configure support for the required LDAP controls on all request processors handling LDAP requests that result from SCIM requests.

About this task

Change the request processor name that was provided as an example and repeat the command for all additional request processors.

Steps

- Use `dsconfig` to change the request processor name (provided as an example) and repeat the command for all additional request processors.

  Use your deployment's request processor name.

  Example:

  ```
  $ 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 PingDirectory 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, the 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

To make the System for Cross-domain Identity Management (SCIM) HTTP connection handler use SSL (which is mandated by the SCIM specification), enable a Key Manager provider and Trust Manager provider.

About this task

> **Note**
>
> This topic applies only to the PingDirectoryProxy server.

Steps

- To enable SSL during the server's initial setup, include the `--ldapsPort` and the `--generateSelfSignedCertificate` subcommands with the `setup` command.
- If your server already has a certificate that you would like to use:

  Choose from:

  - Set the `key-manager-provider` to the value you set when you enabled SSL in the server..
  - Define a new key manager provider.

> **Tip**
>
> Learn more in 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.

   Example:

   ```
   $ 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.

   Example:

```
$ 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`.

   Example:

   ```
   $ 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:

   Example:

   ```
   $ bin/dsconfig set-http-servlet-extension-prop \
     --extension-name SCIM \
     --set oauth-token-handler:ExampleOAuthTokenHandler
   ```

Using HTTP basic authentication with bare UID on the PingDirectoryProxy server

Clients can authenticate to the System for Cross-domain Identity Management (SCIM) extension using HTTP basic authentication and a bare UID value.

About this task

> **Note**
>
> This topic applies only to the PingDirectoryProxy server.

When a SCIM extension is hosted by a PingDirectoryProxy server, you must explicitly configure the server with the names of subordinate base distinguished names (DNs) to search.

> **Note**
>
> Make sure to specify your deployment's subordinate base DN.

Steps

- To configure the server with the names of subordinate base DNs, run the following command on the PingDirectoryProxy server for every base DN that you can access using SCIM.

```
$ 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 PingDirectory server is using a self-signed certificate.

  Example:

```
$ 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 PingDirectory 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:

  Example:

```
$ curl -u "user.0:password" \
  -k "https://localhost:8443/scim/ServiceProviderConfigs"
```

Configuring the Identity Access API

After 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.

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 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.
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`.

   Example:

   ```
   $ 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
   ```

   Result:

   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.

Result:

```
$ 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 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 can 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, see 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
```

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.

Learn more about schema mapping in Mapping the 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 can be present for each resource. Attributes can 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.

Validating the updated SCIM schema

The 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 server, meets the requirements of the SCIM specification regarding resource ID immutability. However, configuring a mapping to the attribute can 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 can 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 can be stored in the server. By default, it is the responsibility of the SCIM client to guarantee ID uniqueness. However, the UID Unique Attribute Plugin can be used by the server to enforce attribute value uniqueness. For information about the UID Unique Attribute Plugin, see Working with the Unique Attribute plugin.

> **Note**
>
> Resource IDs cannot be mapped to virtual attributes. Learn more about configuring SCIM resource IDs in 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 can 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, the 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 PingDirectory server and the PingDirectoryProxy server using the OAuthTokenHandler API (meaning through 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 PingDirectory 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`.

SCIM 2.0 servlet extension configuration

The PingDirectoryProxy server provides a servlet extension that implements version 2 of the System for Cross-domain Identity Management (SCIM) specification.

The SCIM 2.0 extension doesn't replace the existing SCIM 1.1 extension, and there are significant differences in how they're configured.

> **Note**
>
> Configure your SCIM configurations, such as attribute mappings, on the PingDirectoryProxy server instead of on the backend PingDirectory servers.
> There is a known issue with dot notation support for SCIM 2.0 sub-attributes.

The PingDirectoryProxy server supports the following System for Cross-domain Identity Management (SCIM) 2.0 endpoints.

Endpoint	Description	Supported HTTP methods
`/ServiceProviderConfig`	Provides metadata that indicates the server's authentication scheme (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

Endpoint	Description	Supported HTTP methods
`/Schemas`	Lists the SCIM schemas that are configured for use on the 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 the 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
`/<resourceType>/.search`	Lists and filters resources. This is used as an alternative to passing query parameters in the URL.	POST
`/<resourceType>/<resourceId>`	Retrieves a single resource (GET), modifies a single resource (PUT, PATCH), or deletes a single resource (DELETE).	• GET • PUT • PATCH • DELETE

Creating your own SCIM 2.0 application

To create your own System for Cross-domain Identity Management (SCIM) 2.0 application, you must download the SCIM 2 SDK.

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 GitHub.

Authentication requirements for SCIM 2.0 requests

All System for Cross-domain Identity Management (SCIM) requests on the 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 does not process any SCIM requests unless it has at least one `encryption-settings` definition in its `encryption-settings` database. You can create the necessary definitions with the `encryption-settings` command-line tool.

For more information, see Encrypting sensitive data.

> **Note**
>
> You must define encryption settings and ensure they match on the proxy and all backend PingDirectory servers.

Defining permissions for SCIM 2.0 requests

Successfully executing a System for Cross-domain Identity Management (SCIM) request on a server depends on both the configured access control instruction (ACI) in the server and the scopes present in the OAuth bearer token used to authenticate the request.

> **Note**
>
> You must define ACIs on the backend PingDirectory servers. Don't define ACIs on the PingDirectoryProxy server.

The server can authorize SCIM 2.0 requests in one of the following ways:

- Internally, the server processes all SCIM 2.0 requests using the `cn=SCIM2 Servlet,cn=Root DNs,cn=config` service account. Allowing a requested operation depends on the relevant ACIs, which can include both the rights granted to that service account and any rights granted to scopes contained in the OAuth bearer token. The `oauthscope` bind rule is useful because it allows the administrator to use the supplied `0Auth` scopes in ACI logic.
- If user mapping is enabled, the subject of the OAuth bearer token is mapped to an account in the server. In this case, whether a requested operation is allowed depends on the ACIs that apply to the mapped user for the requested operation. You can also use the `oauthscope` bind rule to grant rights based on scopes in the presented token.

Because of implementation details, access to the `objectclass` operational LDAP attribute is necessary for SCIM requests to properly execute. Don't give the service account access to `objectclass` on a global level. Instead, add the ACI granting `objectclass` access to the LDAP subtree to expose to clients.

Learn more in Configuring Permissions for SCIM 2.0 Operations.

> **Note**
>
> ACIs that don't 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 also grants unconditional read access to SCIM requests regardless of the provided OAuth scopes. This is because the requests are processed through the service account.

Enabling user mapping for SCIM 2.0 operations

Access token validator identity mapping ties a local user account to an operation performed in the SCIM 2.0 servlet in the PingDirectory server.

About this task

> **Note**
>
> This topic applies only to the PingDirectory server.

The default configuration for the PingDirectory server for the SCIM 2.0 servlet doesn't require an access token to map to a local user, and operations are recorded in the logs as the `SCIM2 Servlet` user. For more detailed logging and auditing, enable the `map-access-tokens-to-local-users` property to require access tokens to map to a local user.

> **Note**
>
> The users that are being mapped to the access tokens must have the necessary access control rights required to perform the operations that the SCIM 2.0 servlet will invoke on their behalf.
> You should update the authorization server to issue tokens that include the `scim2` scope alongside any other scopes you need for access control purposes.

The `map-access-tokens-to-local-users` property is an optional configuration with the three settings shown in the following table.

Setting	Definition
`Disabled` (default)	The server doesn't attempt to map SCIM 2.0 access tokens to local users and operations are processed under the authority of the `SCIM2 Servlet` user.
`Optional`	The server attempts to map SCIM 2.0 access tokens to local users and, if successful, the operations are processed under the authority of that user. The distinguished name (DN) of the mapped user appears in the access logs. If unsuccessful, the server falls back to the default behavior.
`Required`	The server must map the SCIM 2.0 access token to one local user or the operation is rejected.

To set the `map-access-tokens-to-local-users` property:

Steps

- Run `dsconfig` with the `set-http-servlet-extension-prop` option.

  Choose from:

  - To set the property to `required`, run the following command.

```
dsconfig set-http-servlet-extension-prop \
  --extension-name SCIM2 \
  --set map-access-tokens-to-local-users:required
```

- To set property to `optional`, run the following command.

```
dsconfig set-http-servlet-extension-prop \
  --extension-name SCIM2 \
  --set map-access-tokens-to-local-users:optional
```

- To reset the property to the default setting, `disabled`, run the following command.

```
dsconfig set-http-servlet-extension-prop \
  --extension-name SCIM2 \
  --set map-access-tokens-to-local-users:disabled
```

SCIM 2.0 Components

Unlike the System for Cross-domain Identity Management (SCIM) 1.1 servlet extension, the SCIM 2.0 system is configured through the admin console or with the `dsconfig` command-line tool.

The SCIM 2.0 system consists of the following components:

SCIM resource type

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 available to each resource. If you enable a SCIM resource type, it must have a designated LDAP structural objectclass and an associated base distinguished name (DN).The two types of SCIM resource types, mapping and pass-through, differ based on the definitions of the SCIM schema the resource types use:

LDAP mapping SCIM

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.

LDAP pass-through SCIM

Does not use an explicitly defined SCIM schema. Instead, an implicit schema is generated dynamically based on the underlying LDAP schema. Use a pass-through SCIM resource type when you need to get started quickly

SCIM schemas

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.

SCIM attributes

Defines an attribute that is available under a SCIM schema. The configuration for a SCIM attribute defines its data type, regardless of whether it's required, single-valued, or multivalued.

SCIM sub-attributes

When a SCIM attribute consists of SCIM sub-attributes, it's defined as a complex attribute.

SCIM attribute mappings (mapping resource types only)

Defines the manner in which a SCIM resource type maps the attributes in its SCIM schemas to native LDAP attributes of the PingDirectoryProxy server.

Correlated LDAP Data Views

Allows a single SCIM resource that consists of attributes that are retrieved from multiple LDAP entries. Learn more in Correlated LDAP data views.

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 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.

Correlated LDAP data views share many configuration settings with SCIM resource types. The following exceptions are below:

If the correlated LDAP data view is attached to a pass-through 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 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.

In this example, a new PingDirectory server is set up using custom sample data. When configuring the correlation, administrators should use attributes that are inherently either immutable or non-volatile, such as `uid` or `entryUUID`. This prevents errors produced by a conflict between the values of primary and secondary correlation attributes.

> **Note**
>
> Administrators can make the correlation SCIM attributes immutable by setting the `--set mutability:read-only` property when defining an attribute in the SCIM schema configuration. That way, SCIM requests cannot modify the values of those attributes.

Steps

1. Copy the following text into the server root directory 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 and set up encryption settings. After this is done, 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.

> **Note**
>
> The only meaningful 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. 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, as 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 an LDAP-mapped SCIM resource type

Add a simple mapping System for Cross-domain Identity Management (SCIM) 2.0 resource type backed by the `inetOrgPerson` LDAP objectclass to a PingDirectoryProxy deployment.

About this task

To configure an LDAP mapped SCIM resource type:

Steps

1. Set up the PingDirectory backend server.
2. Export the `encryption-settings` definition with the tool's `export` subcommand.

   Example:

   For this example, use the default settings to use sample data and configured data encryption.

   ```
   encryption-settings export --output-file exported-key
   ```

3. Set up the PingDirectoryProxy server and import the `encryption-settings` definition file that was created in the previous step.
4. To configure the LDAP external server, use the `create-initial-proxy-config` tool.
5. Create the SCIM schema for the resource type to use.

   Example:

```
dsconfig create-scim-schema \
--schema-name urn:pingidentity:schemas:User:1.0 \
--set display-name:User
```

6. 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
```

7. Create the LDAP mapping SCIM resource type on the PingDirectoryProxy server.

Example:

```
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
```

8. To create the SCIM attribute mappings, run the following commands.

```
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
```

9. Configure the SCIM2 HTTP Servlet Extension to use a Mock Access Token Validator.

> **Note**
>
> Never use Mock Access Token Validators 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"
```

10. To confirm that the new resource type is successfully added, send the following request to the PingDirectoryProxy server's SCIM `/ResourceTypes` endpoint.

> **Note**
>
> The HTTP port can vary depending on the deployment configuration.

```
curl -k -X GET \
https://localhost:8443/scim/v2/ResourceTypes \
-H 'Authorization: Bearer {"active":true}'
```

Result:

The following JSON object is displayed 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

Configure permissions so that POST requests with the `userAdd` scope succeed on a PingDirectoryProxy deployment.

Before you begin

Set up an LDAP mapping System for Cross-domain Identity Management (SCIM) 2.0 resource type for the `inetOrgPerson` objectclass.

Learn more in Configuring an LDAP-mapped SCIM resource type.

About this task

To configure permissions:

Steps

1. Set the SCIM resource type property:

   Choose from:

   - If the SCIM resource type being targeted already has a value for the `create-dn-pattern` property, skip to step 2.
   - To set the SCIM resource type property, 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.

```
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"]}'
```

> **Note**
>
> The HTTP port can vary depending on the deployment configuration.

Example:

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. For more information on viewing the full server error message, see Troubleshooting the SCIM 2.0 servlet Extension.

3. **Add an access control instruction (ACI) to the backend server's `ou=People,dc=example,dc=com` subtree.**

    1. **Run the following `ldapmodify` command for creating the ACI on the backend PingDirectory server (and not the PingDirectoryProxy) endpoint.**

    ```
    $ 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";)
    ```

    > **Note**
    >
    > This ACI doesn't grant write access to attributes, which means modify operations will not succeed. For more information on ACI configurations, see Overview of access control.
    > This ACI grants permission to add entries to the specified subtree as long as the SCIM request contains the `userAdd` scope.

4. **Send the POST request to the SCIM `/Users` endpoint again, and include the `userAdd` scope in the bearer token.**

Example:

```
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"]}'
```

Result:

The response from the server contains the created SCIM resource, which also contains 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

You can cap the total number of resources that are matched by a search to prevent exhausting server resources.

The configuration for each System for Cross-domain Identity Management (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:

- The client must refine its search filter to return fewer matches.
- You should configure paged searches as shown in Using paged SCIM searches.

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 reduce memory usage.

Before you begin

> **Note**
>
> The paged System for Cross-domain Identity Management (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 once per PingDirectoryProxy.

> **Tip**
>
> If you're not sure if 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
```

`<proxying-request-processor>` is the request processor handling the entries targeted by the search.

About this task

The PingDirectoryProxy server does SCIM searches using LDAP requests. After you complete the following steps, the 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 can't page the results, the backend server still returns the results. In this case, the PingDirectoryProxy server handles the sorting and paging itself.

If your SCIM searches result in an error because the request matched too many results, you can avoid the error by using paged searches. For more information, see SCIM 2.0 searches.

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`.

   Example:

   Your search might look like the following.

```
https://<proxy-hostname>:<proxy-port>/scim/v2/Users/?filter=st eq "TX"&sortBy=sn&sortOrder=ascending
```

This 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. To find the SCIM resource type, `structural-ldap-objectclass`, `include-base-dn`, and `include-filter` values, run the following 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. For each backend server:

    1. Create a Virtual List View (VLV) index for your search 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 distinguished name (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 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)`.

Option	Description
`--set sort-order`	Specifies whether to sort ascending (+) or descending (-) and the LDAP attribute to sort by. If the SCIM search doesn't specify the `sortBy` parameter, specify the sort order as `+entryUUID`.

Example:

Each SCIM search that you want to produce paged results must have its own VLV index.

Recall the original, decoded SCIM search, shown here.

```
https://<proxy-hostname>:<proxy-port>/scim/v2/Users/?filter=st eq
"TX"&sortBy=sn&sortOrder=ascending
```

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
```

2. Using the command line, stop the server, rebuild the index, start the server, and run the `rebuild-index` command specifying the base DN and the name of the index.

```
$ rebuild-index --baseDN  <baseDN-value>  --index  <name-of-index>
```

Example:

```
$ 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.

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

If the client does not provide values for one of the parameters, the search uses the corresponding default value shown in the table.

SCIM 2.0 PATCH operations

You can use a PATCH request to modify a System for Cross-domain Identity Management (SCIM) 2.0 resource that has one or more required SCIM 2.0 attributes.

The requester needs permissions to read the values of these required attributes and to write permissions for the attributes being modified, even if the PATCH request does not alter the requirements.

Example

You can 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 fails 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"
        }]
                }
```

Troubleshooting the SCIM 2.0 servlet Extension

To troubleshoot the SCIM 2.0 servlet extension, you must enable the Debug Trace Logger.

About this task

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 like the following.

```
{
        "schemas": [
        "urn:ietf:params:scim:api:messages:2.0:Error"
        ],
        "status": "400",
        "detail": "Request failed: correlationID='073eb1a8-8c51-48b3-83a0-380e1d4b4ab9'"
}
```

Steps

- To view these messages, enable the Debug Trace Logger through the admin console or with the following `dsconfig` command.

  Example:

  ```
  dsconfig set-log-publisher-prop --publisher-name "Debug Trace Logger" \
                          --set enabled:true --add scim-message-type:error
  ```

  Result:

  After you enable the Debug Trace Logger, the server begins logging information related to SCIM operations to the `/logs/debug-trace` file, as in the following example.

  ```
  [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 allows for matching the ID in the HTTP responses to the messages in the `debug-trace` log so that 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 System for Cross-domain Identity Management (SCIM) 2 servlet, you can disable the SCIM 2.0 servlet extension.

Steps

1. Remove the SCIM 2 HTTP servlet extension from the HTTPS connection handler as well as from any other HTTP connection handlers.
2. Restart the handler.

Example

```
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
```

Troubleshooting a multiple correlation entry error

When using secondary store adapters in PingAuthorize or correlated Lightweight Directory Access Protocol (LDAP) data views with the System for Cross-domain Identity Management (SCIM) 2 implementation in PingDirectory or PingDirectoryProxy, an error is displayed in the SCIM 2 response to the user request if multiple entries have a secondary correlation attribute value that matches the primary correlation attribute value. The error is applicable to each product.

There are three options for resolving this error:

- Configure one or more correlation attribute pairs on the secondary store adapter or correlated LDAP data view so that for each primary entry, there is only one other entry with matching secondary correlation attribute values. This is the recommended option.
- Modify the server data so that for each primary entry, there is only one other entry with a matching secondary correlation attribute value.

  > **Note**
  >
  > Because this option requires making changes to individual entries, it is not efficient for larger servers.

- Remove the secondary store adapter or correlated LDAP data view.

  > **Important**
  >
  > Because this option involves complete removal of the feature, only do it if the other options do not resolve the error.

Monitoring the PingDirectory Suite of Products

The PingDirectory and PingDirectoryProxy servers provide a flexible monitoring framework that enables you to track server performance.

This framework exposes its monitoring information under the `cn=monitor` entry and provides interfaces through the admin console, SNMP, Java Management Extensions (JMX), and over LDAP. The PingDirectory server also provides a tool, the Periodic Stats Logger, to profile server performance.

The monitor backend

The server exposes its monitoring information under the `cn=monitor` entry.

Administrators can use various means to monitor the servers, including SNMP, the admin console, JConsole, LDAP command-line tools, and the Periodic Stats Logger.

> **Note**
>
> To display server component activity and state, use the **`bin/status`** tool.

To see the list of all monitor entries, use **`ldapsearch`** as in the following example.

```
$ 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.

Component	Description
Active Operations	Provides information about the operations currently being processed by the 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 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. The client connection information contains a name followed by an equal sign and a quoted value, such as `connID="15", connectTime="20100308223038Z"` and so forth.
Connection Handlers	Provides information about the available connection handlers on the server, which includes the LDAP and LDIF connection handlers. These handlers are used to accept client connections 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 server.

Component	Description
General	Provides general information about the state of the server, including product name, vendor name, server version, etc.
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 Java virtual machine (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 several user-defined buckets of information, including the total number of operations processed, overall average response time in milliseconds (ms), number of processing times between 0ms and 1ms, and so forth.
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, Java version, and so forth.
Version	Provides information about the server version, including build ID, version, revision number, etc.
Work Queue	Provides information about the state of the PingDirectory 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, which is 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 the 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 might 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 Java Runtime Environment (JRE) 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 and take additional action if the amount of usable space drops below the defined thresholds.

You can configure three thresholds 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 server generates an administrative alert but remains fully functional. It generates alerts at regular intervals that you configure (such as once a day) unless action is taken to increase the amount of usable space. The server generates additional alerts as the amount of usable space is further reduced, such as 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. If the amount of usable space drops below this threshold, the server generates an alert notification and begins rejecting all operations requested by non-root users with `UNAVAILABLE` results. The server should continue to generate alerts during this time. When the server enters this mode, an administrator has to take some kind of action, such as, running a command to invoke a task or removing a signal file, before the server resumes normal operation. This threshold must be less than or equal to the low space warning threshold. If they are equal, the server begins rejecting requests from non-root users immediately upon detecting low usable disk space.

Out of space error threshold

This threshold can also be defined as a percentage or absolute size. If the amount of usable space drops below this threshold, the server generates a final administrative alert and shuts itself down. This threshold must be less than or equal to the low space error threshold. If they are equal, the server shuts itself down rather than rejecting requests from non-root users.

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 can 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 causes 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 might 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 might become, how much space log files might consume, and how many backups are stored.

The threshold values can be specified either as absolute sizes or as percentages of the total available disk space. You must specify all values 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 does 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 preserves its integrity but might 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 generates periodic warnings about low available disk space but remains fully functional for as long as possible. If all three threshold values are set to zero, then the server does not attempt to warn about or otherwise react to a lack of usable disk space.

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.

> **Note**
>
> This topic applies only to the PingDirectory server.

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 that can be monitored are available in the `cn=monitor` backend.

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 Java Runtime Environment (JRE) 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

Utilization metrics are gathered through 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, such as `eth0` or `lo`, and by disk device names, such as `sd1, sdab, sda2, scsi0`.

Monitoring key performance indicators by application

PingDirectory server can be configured to track many key performance metrics, such as throughput and response-time, by the client applications requesting them.

This feature is invaluable for measuring whether the 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
- Can be accessed in the Periodic Stats Logger
- Can be made available for collection by the Metrics server

For more information on using the Stats Logger, see Profiling server performance using the Stats Logger.

Proxy considerations for tracked applications

In a proxy environment, the criteria should be defined in the PingDirectoryProxy server, because the PingDirectoryProxy server passes the application name through to the PingDirectory server in the intermediate client control.

If a client of the PingDirectoryProxy server or the PingDirectory server happens to use the intermediate client control, then the client name specified in the control is used as the application name regardless of the criteria listed in the `tracked-application` property.

Monitoring using SNMP

The PingDirectory server supports real-time monitoring using SNMP. The server provides an embedded SNMPv3 subagent plugin that, when enabled, sets up the server as a managed device and exchanges monitoring information with a primary agent based on the AgentX protocol.

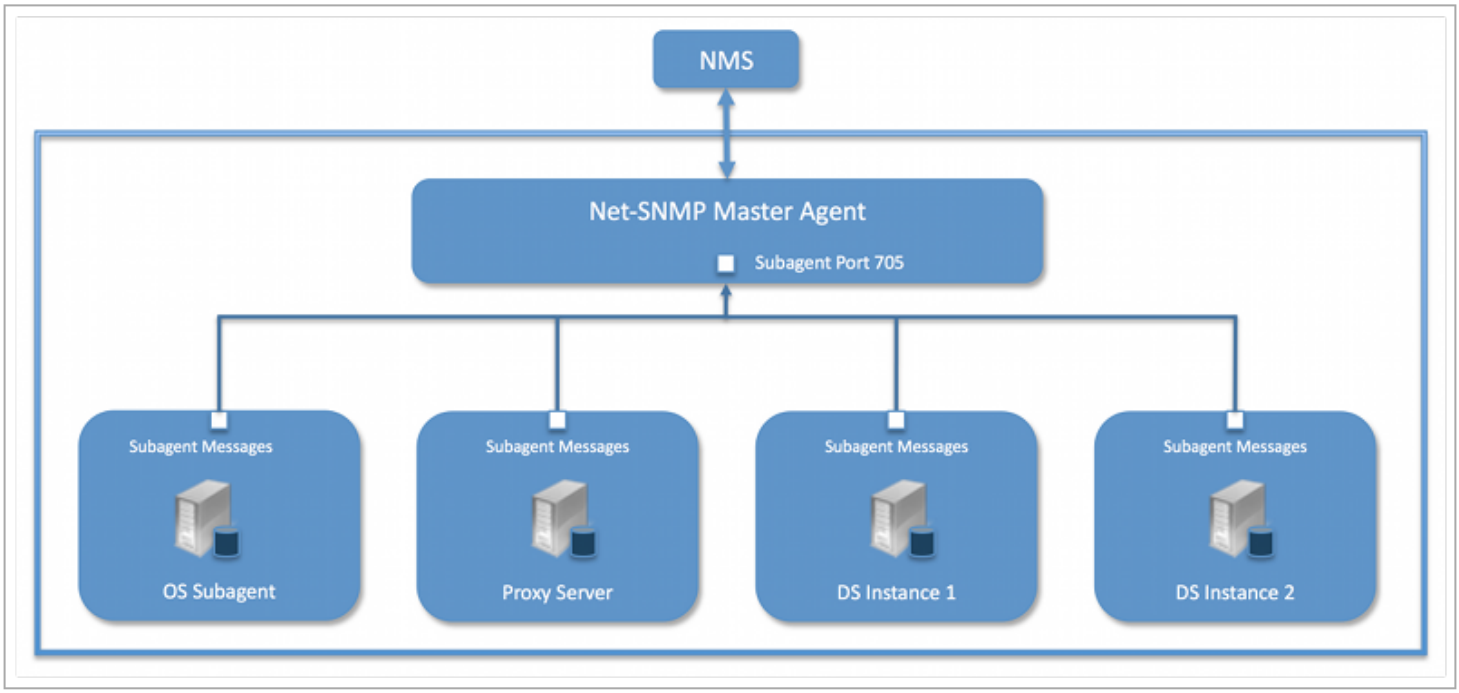
> **Important**
>
> SNMP is deprecated.

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 primary agent, whose main responsibility is to translate the SNMP protocol messages and multiplex any request messages to the subagent on each managed device, such as a PingDirectory server instance, a PingDirectoryProxy server, a PingDataSync server, or an OS Subagent. The primary agent also processes responses or traps from the agents. Many vendors provide commercial NMS systems. Consult with your NMS system for specific information.

The server contains an SNMP subagent plugin that connects to a Net-SNMP primary agent over TCP. The main configuration properties of the plugin are the address and port of the primary 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 management information base (MIB) server with the server instance. After the plugin's startup method is called, it starts a session thread with the primary agent. Whenever the connection is lost, the subagent automatically attempts to reconnect with the primary agent. The 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 through an NMS system.



Example SNMP Deployment

Note

The PingDirectory server was designed to interface with a Net-SNMP (5.3.2.2 or later) primary agent implementation with AgentX over TCP. Many operating systems provide their own Net-SNMP module. However, Service Management Automation (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 primary agent that only enables UNIX Domain Sockets, you should download and install a separate Net-SNMP binary from its website.

Configuring SNMP

Because all server instances provide information for a common set of management information bases (MIBs), each server instance provides its information under a unique SNMPv3 context name equal to the server instance name.

About this task

The server instance name is defined in the Global Configuration and is constructed from the host name and the server LDAP port by default. Information must be requested using SNMPv3, specifying the context name that pertains to the desired server instance.

> **Note**
>
> The 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. **To enable the server's SNMP plugin, use the `dsconfig` tool.**

   > **Note**
   >
   > The SNMPv3 context name is limited to 30 bytes maximum. Any context name longer than 30 characters returns an error message when you attempt to enable the plugin.
   > The default context server name is the server instance name and the LDAP port number, so take note of the length of the fully-qualified DNS host name.

   1. **Specify the address and port of the SNMP primary agent.**
   2. **On each server instance, enable the SNMP subagent.**
   3. **Enable the SNMP Subagent Alert Handler so that the sub-agent sends traps for administrative alerts generated by the server.**

      Example:

      ```
      $ bin/dsconfig set-alert-handler-prop \
        --handler-name "SNMP Subagent Alert Handler" --set enabled:true
      ```

2. **View the error log.**

   Result:

   A message displays that the primary 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 `snmpd.conf` configuration file and add the directive to run the agent as an AgentX primary agent.**

   The file is often located in `/etc/snmp/snmpd.conf`.

Example:

```
master agentx agentXSocket tcp:localhost:705
```

> **Note**
>
> Using `localhost` means that only sub-agents running on the same host can connect to the primary agent. This is necessary because 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.

   Example:

   ```
   trap2sink localhost public 162
   ```

5. To create a SNMPv3 user, add the following lines to the `/etc/snmp/snmpd.conf` file.

   Example:

   ```
   rwuser initial
   createUser initial MD5 setup_passphrase DES
   ```

6. To create the SNMPv3 user, run `snmpusm`.

   Example:

   ```
   snmpusm -v3 -u initial -n "" -l authNoPriv -a MD5 -A setup_passphrase \
   localhost create snmpuser initial
   ```

7. Start the `snmpd` daemon.

   Result:

   A message displays in the server's error log.

   ```
   The SNMP subagent connected successfully to the master agent
   at localhost:705. The SNMP context name is host.example.com:389
   ```

8. To see the alerts that are generated by the server, set up a trap client.

   1. Create a config file in `/tmp/snmptrapd.conf`.
   2. Add the `authcommunity log, execute public` directive to the file.

      > **Note**
      >
      > The directive specifies that the trap client can process traps using the public community string and can log and trigger executable actions.

9. Install the MIB definitions for the Net-SNMP client tools in the `/usr/share/snmp/mibs` directory.

Example:

```
$ cp resource/mib/* /usr/share/snmp/mibs
```

10. To run the trap client, run the `snmptrapd` command.

    Example:

    This example specifies that the command should not create a new process using `fork()` from the calling shell (`-f`), should not read any configuration files (`-C`) except the one specified with the `-c` option, should print to standard output (`-Lo`), and then specifies 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 (`server-root/resource/mib`).

    ```
    $ snmptrapd -f -C -c /tmp/snmptrapd.conf -Lf /root/trap.log -Dusm \
      -m all -M +/usr/share/snmp/mibs
    ```

11. To test the feature, run the Net-SNMP client tools.

    You must use the following options:

    - `-v` *<SNMP version>*
    - `-u` *<user name>*
    - `-l` *<security level>*
    - `-n` *<context name (instance name)>*
    - `-A` *<user password>*

      Example:

      In this example, 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 server provides SMIv2-compliant management information base (MIB) definitions (RFC 2578, 2579, 2580) for distinct monitoring statistics. These MIB definition text files are in the server's `/resource/mib` 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 server connects to and statistics about the operations invoked by the 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 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 that can help diagnose how much outstanding work replication might 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 PingDirectory server and the PingDirectoryProxy server, see the text files located in the server's `/resource/mib` directory.

The 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 admin console

The admin console can be used to monitor items, such as disk space usage, active operations in the server, and alarms raised.

About this task

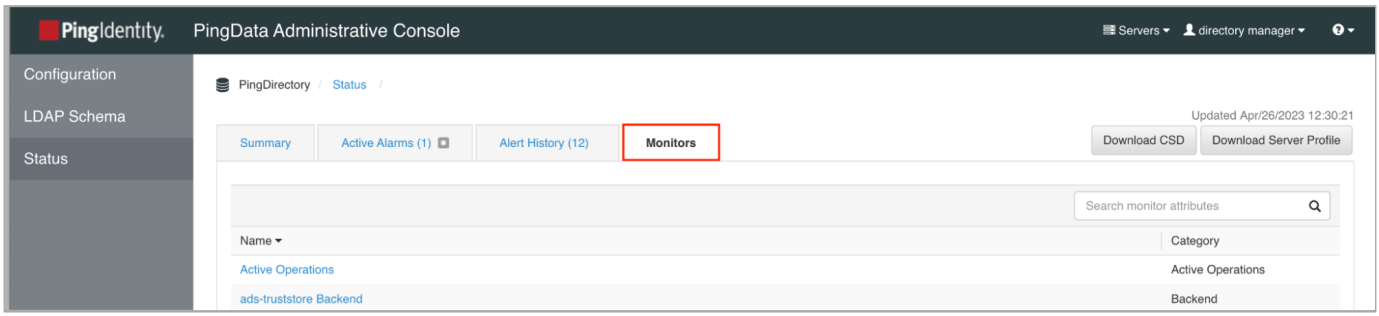
The console provides a status option that accesses the server's monitor content.

To view the monitor dashboard:

Steps

1. In your browser, go to http://<*server-name*>:<*port*>/console.
2. Enter the root user distinguished name and password. Click Login.
3. In the top-level navigation menu, select Status.

4. Click the Monitors tab.



Result

The Monitors page opens, and you can monitor your server by selecting a monitor attribute.

Accessing the Processing Time Histogram

The PingDirectory server provides a processing time histogram that classifies operation response time into user-defined buckets.

About this task

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`
- `modifyDN`
- `search`

Steps

1. From the admin console, go to Configuration > Status > Monitors.
2. Select Processing Time Histogram.

> **Note**
>
> You can access other monitor entries in similar ways.

Monitoring with JMX

The PingDirectory server supports monitoring the Java virtual machine (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. In addition to the monitor information that the server provides, you can monitor generic metrics exposed by the JVM, including:

- Memory pools
- Threads
- Loaded classes
- MBeans

You can also subscribe to receive JMX notifications for any administrative alerts that are generated within the server.

Running JConsole

Run `jconsole` to monitor the memory usage and thread activity of a Java virtual machine.

Before you begin

Before you can run `jconsole`, you must:

- Configure and enable the Java Management Extensions (JMX) Connection Handler for the server using the `dsconfig` tool.

  > **Note**
  >
  > For more information, see Configuring the JMX connection handler and alert handler.

- Invoke the `jconsole` executable by entering `jconsole` in your command-line interface or terminal.

  > **Note**
  >
  > If *<JDK_HOME>* is not set in your path, you can access `jconsole` in the `bin` directory of the `<JDK_HOME>` path.

Steps

1. To open the Java Monitoring & Management Console, run `jconsole`:

   Choose from:

   - To monitor a specific process ID for your application, run `jconsole <process ID>`.
   - To run `jconsole` remotely, run `jconsole <hostname:port>`.

     If SSL is configured on the JMX Connection Handler, you must specify the PingDirectory server `.jar` file in the class path when running `jconsole` over SSL, as in the following example:

```
$ jconsole -J-Dcom.unboundid.directory.server.protocol.jmx.trustStorePath=$INSTANCE_ROOT/config/
truststore -J-Dcom.unboundid.directory.server.protocol.jmx.trustStorePin=secret -J-
Dcom.unboundid.directory.server.protocol.jmx.trustStoreType=JKS -J-classpath -J"$INSTANCE_ROOT/lib/*:/
Library/Java/JavaVirtualMachines /jdk1.8.0_201.jdk/Contents/Home/lib/jconsole.jar"
```

Set the following properties in the above command:

- Set the `com.unboundid.directory.server.protocol.jmx.trustStorePath` property with the full trust store path.
- Set the `com.unboundid.directory.server.protocol.jmx.trustStoreType` property if the default type, Java KeyStore (JKS), is not used.
- Set the `com.unboundid.directory.server.protocol.jmx.trustStorePin` property with the trust store file password, if there is one.
- Set the `com.unboundid.directory.server.protocol.jmx.trustStorePinFile` property with the file path containing the trust store password in plain text.

  Do not use this property if the `com.unboundid.directory.server.protocol.jmx.trustStorePin` property is used.

> **Note**
>
> When establishing a connection to **`jconsole`**, a window opens asking if you want to retry connecting insecurely. Click **Insecure connection**.
> Although **`jconsole`** considers anything other than `SslRmiClientSocketFactory` to be insecure, choosing to retry insecurely will enable PingDirectory's secure client socket factory.

2. In the Java Monitoring & Administrative Console window, click Local Process, and then click the PID corresponding to the server.
3. Review the resource monitoring information.

Monitoring the server using JConsole

Set up JConsole to monitor the server using a remote process.

Steps

1. Start the server.

   Example:

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

2. Enable the Java Management Extensions (JMX) Connection handler using **`dsconfig`** with the `set-connection-handler-prop` option.

> **Note**
>
> The handler is disabled by default.
> Include the LDAP connection parameters, such as host name, port, bindDN, and bindPassword.

Example:

```
$ bin/dsconfig set-connection-handler-prop \
  --handler-name "JMX Connection Handler" --set enabled:true
```

3. Assign privileges to a regular user account with the `ldapmodify` tool.

> **Warning**
>
> Do not use a root user account, as using a root user account would be a security risk.

Example:

This example grants `jmx-read`, `jmx-write`, and `jmx-notify` privileges 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. In Java Monitoring & Administrative Console, click Remote Process, and enter the following JMX URL using the host and port of your server.

```
service:jmx:rmi:///jndi/rmi://<host>:<port>/com.unboundid.directory.server.protocols.jmx.client-unknown
```

5. In the Username and Password fields, enter the bind distinguished name (DN) and password for a user that has at least the `jmx-read` privilege.
6. Click Connect.
7. Click com.unboundid.directory.server, and expand the `rootDSE` node and the `cn-monitor` sub-node.
8. Select a monitoring entry.

Monitoring using the LDAP SDK

You can use the monitoring API to retrieve monitor entries from the PingDirectoryProxy server and retrieve specific types of monitor entries.

For example, you can retrieve all monitor entries published by the 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 repository on GitHub.

Monitoring over LDAP

The PingDirectory server exposes a majority of its information under the `cn=monitor` entry.

Steps

- To access these entries over LDAP, use the `ldapsearch` tool.

  Example:

  ```
  $ 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 PingDirectory server includes 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 `.csv` file, 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 value of less than 1 second. You can customize the statistics logged and their verbosity.

The Stats Logger can also be used to view historical information about server statistics, including:

- Replication
- LDAP operations

- Host information
- Gauges

To update the configuration of the existing Stats Logger Plugin, you can either:

- Set the advanced `gauge-info` property to `basic/extended` to include this information.
- Create a dedicated Periodic Stats Logger for information about statistics of interest.

Learn more in Enabling the Stats Logger.

Enabling the Stats Logger

About this task

By default, the PingDirectory server's built-in Stats Logger is disabled. To enable the Stats Logger:

Steps

1. Run `dsconfig` in interactive mode.

   When you are prompted, enter the LDAP or LDAPS connection parameters.

   Example:

   ```
   $ bin/dsconfig
   ```

2. To change to the `Advanced Objects` menu, enter `o`.
3. On the main menu, enter the selection for `Plugins`.
4. On the `Plugin` menu, enter the selection for `View and edit an existing plugin`.
5. In the `Plugin` selection list, enter the selection for `Stats Logger`.
6. On the `Stats Logger Plugin` menu, enter the selection to set the `enabled` property to `TRUE`. To save and apply the configuration, enter `f`.

   Result:

   The default logger logs information about the server every second to `<server-root>/logs/dsstats.csv`. If the server is idle, nothing is logged.

   > **Note**
   >
   > To change the idle logging behavior, set the `suppress-if-idle` property to `FALSE`.

7. Run the PingDirectory server.

   Example:

   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.

> **Tip**
>
> To see an example command, run `search-and-mod-rate --help`.

8. View the Stats Logger output at `<server-root>/logs/dsstats.csv`.

   You can open the file in a spreadsheet.

Configuring multiple Periodic Stats Loggers

About this task

You can create multiple Periodic Stats Loggers to:

- Log different statistics
- View historical information about gauges
- 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` in interactive mode.

   When prompted, enter the LDAP or LDAPS connection parameters.

   Example:

   ```
   $ bin/dsconfig
   ```

2. To change to the `Advanced Objects` menu, enter `o`.
3. In the main menu, enter the selection for `Plugins`.
4. In the `Plugin management` menu, enter the selection for `Create a new plugin`.
5. To use an existing plugin as a template, in the `Create a New Periodic Stats Logger Plugin` menu, enter `t`.
6. Enter the number corresponding to the existing stats logger as a template.
7. Enter a descriptive name for the new stats logger.

   For this example, enter `Stats Logger-10s`.

8. Enter the log file path to the file.

   For this example, enter `logs/dsstats2.csv`.

9. In the menu, make any other changes to the logger. To save and apply the configuration, enter `f`.

   Example:

   For this example, change the `log-interval` to `10s`, and the `suppress-if-idle` to `FALSE`.

Result

You should 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` with 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. New Custom Logged Stats are automatically included in the Periodic Stats Logger output.

In addition to allowing a straight pass-through of the values using the raw statistic-type, you can configure attributes to be treated as any of the following:

- A counter, where the interval includes the difference in the value since the last interval
- An average value held by the attribute during the specified interval
- A minimum value held by the attribute during the specified interval
- 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 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 show you how to configure a Custom Logged Statistics Logger, the following sections reproduce the built-in `Consumer Total GB` column that displays 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, when prompted, enter the LDAP or LDAPS connection parameters.

Example:

```
$ bin/dsconfig
```

2. In the server's `Advanced Objects menu`, enter the selection for `Custom Logged Stats`.
3. In the `Custom Logged Stats menu`, enter the selection for `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. In the `monitor-objectclass` property menu, enter the `objectclass` attribute to monitor.

   For this example, enter `ds-memory-usage-monitor-entry`.

   > **Note**
   >
   > To view the entry, run **`ldapsearch`** using the base distinguished name (DN) `cn=JVM Memory Usage,cn=monitor` entry.
7. Enter the attributes of the monitor entry that you want to log in the stats logger, and then press Enter again to continue.

   For this example, enter `total-bytes-used-by-memory-consumers`.
8. Specify the type of statistics for the monitored attribute to appear in the log file.

   For this example, enter the option for `raw statistics` as recorded by the logger.
9. In the `Custom Logged Stats menu,` review the configuration.
10. To set up a column name that lists the memory usage, enter the option to change the `column-name` property.
11. To add a specific label for the column name, enter the option to `add a value`, and then enter `Memory Consumer Total (GB)`. To continue, press Enter.
12. Confirm that you want to use the `column-name` value that you entered in the previous step. To use the value, press Enter.
13. To scale the `Memory Consumer Totals (GB)` by 1 gigabyte (GB), in the `Custom Logged Stats menu`, enter the option to change the `divide-value-by` property.
14. In the `divide-value-by` property menu, enter the option to change the value, and then enter `1073741824`.

    1073741824 bytes is equivalent to 1 gigabyte.
15. In the `Custom Logged Stats menu`, review your configuration. To save and apply the settings, enter `f`.

    Example:

```
>>>> 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]:
```

Result:

A message of `The Custom Logged Stats was created successfully` is returned.

After 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

About this task

The following task replicates the previous procedure using `dsconfig` in non-interactive mode.

To create a custom stats logger:

Steps

- Run the `dsconfig` non-interactive command with the `create-custom-logged-stats` option.

  Example:

  In this example, the 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
```

Enabling and configuring the 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.

About this task

You can configure the Monitoring Endpoint using the `dsconfig` command or the admin console.

Steps

- To create the StatsD monitoring Endpoint, use either of the following:

  Choose from:

  - To use the command-line, run `dsconfig` with the `create-monitoring-endpoint` option.

    This example configures 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
    ```

  - To use the admin console:

    1. From the admin console, click Show Advanced Configuration.
    2. In the Logging, Monitoring, and Notifications section, click Monitoring Endpoints.
    3. Click New Monitoring Endpoint.

  When you configure Monitoring Endpoint include:

  - The endpoint's host name
  - The endpoint's port
  - A toggle to use TCP or UDP
  - A toggle to use SSL if you use TCP

You can configure a StatsD Monitoring Endpoint with custom tags using the `additional-tags` property. This adds the defined tags to each metric message sent to the endpoint. Each tag should be created in a "key=value" format. Additional tags are appended to the end of the StatsD message. Here is a sample StatsD message with custom tags:

```
example.metric:123|g|#tag1:value1,tag2:value2
```

> **Note**
>
> You can send data to any number of monitoring endpoints.

Enabling and configuring the Stats Collector Plugin

The Stats Collector Plugin controls the metrics used by the StatsD monitoring endpoint.

About this task

> **Note**
>
> This topic applies only to the PingDirectory server.

To send metrics with the StatsD monitoring endpoint, enable the Stats Collector Plugin and configure the plugin to indicate which metrics to send.

Examples of metrics you can send are:

- Busy worker thread count
- Garbage collection statistics
- Host system metrics such as CPU and memory

> **Tip**
>
> For a list of available metrics, use the interactive **`dsconfig`** menu for the Stats Collector Plugin, or in the admin console, edit the Stats Collector plugin, as explained in the second example.

Steps

- To enable and configure the Stats Collector Plugin, use either of the following:

  Choose from:

  - To use the command line, run `dsconfig` with the `set-plugin-prop` option.

    This example enables the Stats Collector Plugin to send host CPU metrics, 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
```

> **Note**
>
> If you are not using Data Metrics Server to monitor your server, you can disable the generation of some unnecessary metrics files for the StatsD Monitoring Endpoint. To do this, set the `generate-collector-files` property on the Stats Collector Plugin to `false`.

- To use the admin console:

  1. In the admin console, click Show Advanced Configuration.
  2. In the LDAP (Administration and Monitoring) section, click Plugin Root.
  3. Edit the Stats Collector plugin and select the configuration options to indicate which metrics to send.

Next steps

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.
- Configure other tools that support StatsD, such as CloudWatch or a Prometheus StatsD exporter, to use the data. You can find more information about this configuration in your tool's StatsD documentation.

Sending metrics to Splunk with StatsD

About this task

> **Note**
>
> This topic applies only to the PingDirectoryProxy server.

Using the StatsD Endpoint type, you can send metric data to a Splunk installation. In Splunk, you can use Secure Sockets Layer (SSL) to secure ports opened for StatsD. You can configure open UDP or TCP ports in Splunk to accept only connections from a certain hostname or IP address.

> **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 securely send UDP or TCP data to Splunk:

Steps

1. Send the data to a Splunk Universal Forwarder.

2. Request that the forwarder use SSL to communicate with the Splunk Indexer.

Troubleshooting the PingDirectory Suite of Products

The PingDirectory Suite of Products provides a highly-reliable service that satisfies your company's objectives. However, if problems do arise, 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 section 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:

Troubleshooting the PingDirectory and PingDirectoryProxy servers

There are several ways to troubleshoot issues with the PingDirectory and PingDirectoryProxy servers.

This section contains the following topics:

- Server gauges
- Working with the collect-support-data tool
- PingDirectory server troubleshooting information
- Monitor entries
- Server troubleshooting tools
- Troubleshooting resources for Java applications

Server gauges

The server provides several built-in gauges to monitor server performance. These gauges are listed in the following table.

> **Note**
>
> Some of the gauges described in the following table apply only to the PingDirectory server, and some apply only to the PingDirectoryProxy server. These gauges are indicated as such. The remaining gauges apply to both servers.

Gauge Name	Enabled by default?	Description
Active Cleaner Threads (Percent)	true	PingDirectory server only: Monitors the percentage of database cleaner threads that are active in a Berkeley DB environment. The resource identifier for this gauge is a backend ID. Use the `list-backends` command to see a list of backend IDs. A separate gauge monitor entry will be created for each monitored backend. Backends can be included or excluded from monitoring by specifying its backend ID in the include resource and exclude resource properties respectively. To keep the database from growing on disk, database cleaner threads copy database information from older, mostly obsolete database files to new database files. At a single point in time, 100% of the cleaner threads might be active, but when averaged over time, the percentage of active cleaner threads should remain relatively low. Even environments that sustain a very high write load do not typically see an average cleaner percent busy over 50%. If the percentage exceeds 90% for over an hour, it is a sign that the database cleaner is not progressing and the act of cleaning is producing as much garbage as it cleans. This typically occurs in environments that are not fully cached, and have not given sufficient memory to the database cache. Either increase the memory available to the database cache (by increasing the JVM size or increasing the `db-cache-percent` setting on the backend) or reduce the `db-cleaner-min-utilization` setting, which reduces the burden on the cleaner thread(s).
Authentication Failure Rate	false	Rate of LDAP bind failures per second. A high rate of failures might indicate a misconfigured client application or a malicious attack.
Available File Descriptors	true	Monitors the number of file descriptors available to the server process. The server allows for an unlimited number of connections by default, but is restricted by the file descriptor limit on the operating system. 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. Running out of available file descriptors can lead to unpredictable behavior and severe system instability.
Certificate Expiration (Days)	true	Monitors the expiration dates of key server certificates. A server certificate expiring can cause server unavailability, degradation, or loss of key server functionality. Certificates nearing the end of their validity should be replaced as soon as possible. See the status tool, or Status in the admin console, for more information about server certificates and how they are managed.

Gauge Name	Enabled by default?	Description
Changelog Database Target Size (Percent)	true	PingDirectory server only: Monitors the size of a changelog database on disk relative to the configured `target-database-size` value for the Replication Server changelog and the LDAP Changelog Backend (cn=changelog). The resource identifier indicates the changelog environment that is being monitored. The server aims to keep the disk usage of the changelog to between 95% and 100% of the target-database-size value. The most common reason that the server exceeds this limit is that there are no more changes that are old enough to purge. This is controlled with the replication-purge-delay setting on the Replication Server configuration object and the `changelog-maximum-age` setting on the 'changelog' Backend configuration object. If this is the case, the effective-purge-delay monitor attribute will match the configured purge delay. To eliminate the alarm in this case reduce the purge delay or increase the target-database-size value. Another reason that the disk usage could exceed this limit is that an export-ldif or online backup of the corresponding backend is running since database files on disk cannot be deleted while this operation is in progress. In practice, this will not occur unless the purge delay and target-database-size setting are configured to very small values and/or the backup is throttled with the `--maxMegabytesPerSecond` option so that it takes an especially long time to complete. The final reason that the limit could be exceeded is that the target-database-size setting is unreasonably small like less than one gigabyte.
Cleaner Backlog (Number Of Files)	true	PingDirectory server only: Monitors the cleaner backlog in a Berkeley DB environment. The resource identifier for this gauge is a backend ID. Use the `list-backends` command to see a list of backend IDs. A separate gauge monitor entry will be created for each monitored backend. Backends can be included or excluded from monitoring by specifying its backend ID in the include resource and exclude resource properties respectively. The backlog is the number of database files that need to be cleaned to reach the target level of utilization. The value over time should stay close to zero. If it remains high or continues to grow, then consider updating the backend configuration to increase the `db-num-cleaner-threads` setting or reduce the `db-cleaner-min-utilization` setting. Temporary spikes in cleaner backlog are common during rebuild-index, export-ldif, and backup operations. The backlog should decrease automatically when these commands complete.
CPU Usage (Percent)	true	Monitors server CPU use and provides an averaged percentage for the interval defined. The monitored resource is the host system's CPU, which does not include a resource identifier. 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.

Gauge Name	Enabled by default?	Description
Database Cache Full (Percent)	true	PingDirectory server only: Monitors the percentage capacity of the database cache currently populated with entries, per backend. The resource identifier for this gauge is a backend ID. Use the `list-backends` command to see a list of backend IDs. A separate gauge monitor entry will be created for each monitored backend. Backends can be included or excluded from monitoring by specifying its backend ID in the include-resource and exclude-resource properties respectively. Server performance can drop off dramatically when the database cache can no longer hold the entire data set. Most deployments are designed to keep the database cache usage comfortably within the configured limits. If this server is intentionally disk-bound, this gauge should be disabled.
DB on Disk to DB Cache Size Ratio	true	PingDirectory server only: Monitors the ratio between the sizes of the on-disk database and the in-memory cache database. This gauge relies on the `db-on-disk-to-db-cache-size-ratio` attribute within database environment monitor entries. The ratio can help when troubleshooting performance issues caused by excessive thrashing between memory and disk. If the ratio grows to a size eight times larger than what can be stored in memory, which is the default threshold, there is an increased risk of degraded performance, and the gauge raises an alert. To resolve this issue, consider increasing the amount of RAM on the system or using entry-balancing with the PingDirectoryProxy server. **Note** The alert threshold for the ratio size is configurable and could be larger or smaller than the default.
Data Set Availability	true	PingDirectoryProxy server only: For each data set, indicates whether there is at least one backend server available to fulfill requests. The resource identifier for this gauge is the name of the load-balancing algorithm used to define the set of backend servers fulfilling requests for the data set. Use the `dsconfig list-load-balancing-algorithms` command to see a list of load balancing algorithms. A data set can be included or excluded from monitoring by specifying its load-balancing algorithm name in the include resource and exclude resource properties respectively. A data set's availability depends upon the availability of the backend servers available to fulfill the PingDirectory server requests. The set of backend servers is defined by the set of servers configured in a data set's load-balancing algorithm. Make sure that backend servers are available to fulfill proxy requests.

Gauge Name	Enabled by default?	Description
Data Set Local Availability	true	PingDirectoryProxy server only: For each data set, indicates whether there is at least one local backend server available to fulfill requests. The resource identifier for this gauge is the name of the load-balancing algorithm used to define the set of backend servers fulfilling requests for the data set. Use the `dsconfig list-load-balancing-algorithms` command to see a list of load balancing algorithms. A data set can be included or excluded from monitoring by specifying its load-balancing algorithm name in the `include resource` and `exclude resource` properties respectively. A data set's availability depends upon the availability of the backend servers available to fulfill the PingDirectory server requests. The set of backend servers is defined by the set of servers configured in a data set's load-balancing algorithm. Locations are used to indicate the set of preferred backend servers that should be used to fulfill requests. You can denote that a backend server is local by setting its external server configuration object's location attribute to be the same as this Directory Proxy Server instance's location set in its global configuration. Make sure that backend servers are available to fulfill proxy requests.
Disk Busy (Percent)	true	Monitors disk busy percentage over the update interval. This gauge requires that the Host System Monitor Provider be enabled and that any monitored disks be registered using the disk-devices property of that Monitor Provider. The resource identifier for this gauge is the disk device name. Use the `iostat` command or a similar system utility to see a list of disk device names. A separate gauge monitor entry will be created for each monitored disk. High disk usage might be indicative of a directory server whose database cache and/or JVM heap size are not large enough to contain the entire data set.
HTTP Processing (Percent)	true	Monitors the percentage of time that request handler threads spend processing HTTP requests. This percentage represents the inverse of the server's ability to handle new requests without queueing.
JVM Memory Usage (Percent)	true	Monitors the percentage of Java Virtual Machine memory that is in use. This value naturally fluctuates because of garbage collection, so the minimum value within an interval is reported since it is a better indication of overall memory growth. When the memory usage exceeds 90%, this should be reported to customer support since the server is either misconfigured or has a memory leak. As memory usage approaches 100%, the server is more and more likely to experience garbage collection pauses, which leave the server unresponsive for a long time. Restarting the server is likely the only remedy for this situation. Before restarting the server, run collect-support-data and capture the output of 'jmap -histo ' to provide to customer support. The pid of the server can be found from `/logs/server.pid`.

Gauge Name	Enabled by default?	Description
LDAP Operation Average Response Time (Milliseconds)	false	Monitors the average response time across all LDAP operations processed by this server since it was started. There is no resource identifier associated with this gauge. The monitored resource is overall response time of all LDAP operations processed by this server since it was started. High response times can be indicative of several factors including a disk-bound server, network latency, or misconfiguration. Enabling the Stats Logger plugin might help isolate problems. See the administration guide for more information on common problems and solutions.
LDAP Operations Failed (Percent)	false	Monitors the percentage of all LDAP operations processed by this server that have failed since it was started. There is no resource identifier associated with this gauge. The monitored resource is overall number of failed LDAP operations processed by this server since it was started. A high percentage of failed operations might indicate misconfiguration of a client or server in a topology.
License Expiration (Days)	true	Monitors the expiration date of the product license. An expired license will cause warnings to appear in the server's logs and in the status tool output. Request a license key through the Ping Identity licensing website or contact sales@pingidentity.com. Use the dsconfig tool to update the License configuration's license key property.
Memory Usage (Percent)	false	Monitors the percentage of memory use averaged over the update interval defined. The monitored resource is the host system's memory use, which does not have a resource identifier. Some operating systems, including Linux, use the majority of memory for file system cache, which is freed as applications need it. If memory use is high, check the applications that are running on the server.
Purge Expired Data Backlog (Number of Entries)	true	PingDirectory server only: Monitors the backlog of entries that need to be purged by a Purge Expired Data Plugin. The resource identifier for this gauge is the name of the configured plugin. Increasing the max-updates-per-second configuration property on the plugin can increase the rate that the plugin purges expired data. It might also be necessary to refine the search that the plugin performs to be more efficient.
Recent Changes Database-to-JVM Heap Size Ratio (Percent)	true	PingDirectory server only: The recent changes database keeps several recent changes made by update operations in changelog change entry form. This database is used by replication and the changelog backends, and unexpected growth affects server start time, memory consumption and space on disk. The ratio of the recent changes database to the overall heap size is used to track the impact on memory consumption. If you expect large changes and the server isn't experiencing issues, increase the alarm threshold. After processing large changes the alarm should clear itself, if not, it might necessary to perform an export and re-import of the data to resolve the issue.

Gauge Name	Enabled by default?	Description
Replication Conflict Growth Rate	false	PingDirectory server only: Growth rate of the number of unresolved conflicts per second over the update interval. The resource identifier for this gauge is the base DN of the replica. Use the `dsreplication status` command to see a list of replicated base DNs. A separate gauge monitor entry will be created for each monitored replica. Replicas can be included or excluded from monitoring by specifying a replication base DN in the include resource and exclude resource properties respectively. Updates to directory server entries in a replication topology can happen independently, since replication guarantees only eventual consistency, not strong consistency. The eventual consistency model 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. However, in some scenarios, manual administrative action is required. Attention should be paid to the origin of client write requests to prevent conflicts.
Replication Connection Status	false	PingDirectory server only: Indicates the connection status of remote replication servers this servers replication topology. The 'cn=schema' backend as well as the local replication server, are excluded using the include-filter property. For all other remote servers in the topology, separate monitor entries will be created per server and replication base DN. The resource identifier for this gauge is a concatenated string containing the data set name, the host and port of the replication server and the replication server ID. A replicated data set depends upon the availability of servers replicating the data. So long as there are other replicas available in a replication topology, a single replica being unavailable should not affect the overall performance of the replication topology. However, unavailable replicas can increase the likelihood of data loss or performance degradation.
Replication Latency (Milliseconds)	false	PingDirectory server only: Average amount of time it takes a modification on one replica to propagate and commit on another replica, in milliseconds. The resource identifier for this gauge is the base DN of the replica. Use the `dsreplication status` command to see a list of replicated base DNs. A separate gauge monitor entry will be created for each monitored replica. Replicas can be included or excluded from monitoring by specifying its base DN in the include-resource and exclude-resource properties respectively. Replication latency can be reported as high after the server starts, if the global configuration property startup-min-replication-backlog-count is not set. That property limits the number of outstanding changes any replica can have before the server will complete the startup process and begin accepting connections. This limits how out of the sync the server is at start up.

Gauge Name	Enabled by default?	Description
Replication Purge Delay (Hours)	true	PingDirectory server only: For the replication server indicates the effective purge delay. In order to protect against missing changes the effective purge delay should be large enough to accommodate servers that have been offline as well as the need to restore from backups.
Replication Servers Available	false	PingDirectory server only: Strong Encryption Not Available
Strong Encryption Not Available	true	The JVM does not appear to support strong encryption algorithms, like 256-bit AES. The server will fall back to using weaker algorithms, like 128-bit AES. To enable support for strong encryption, update your JVM to a newer version that supports it by default, or install or enable the unlimited encryption strength jurisdiction policy files in your Java installation.
Undeletable Database Files (Percent)	true	PingDirectory server only: Monitors the percentage of undeletable database files in a Berkeley DB environment. The resource identifier for this gauge is a backend ID. Use the `list-backends` command to see a list of backend IDs. A separate gauge monitor entry will be created for each monitored backend. Backends can be included or excluded from monitoring by specifying its backend ID in the include resource and exclude resource properties respectively. The percentage of undeletable database files tracks the percentage of database files that have been cleaned but cannot be deleted because they are being used by a database maintenance operation. This could be a separate, offline process such as export-ldif or verify-index, or it could be a task running within the server such as a backup or replication initialization. A small percentage of undeletable files is expected when these commands are running, but a high percentage could indicate that one of these operations is having problems and should be canceled and restarted to avoid the database growing too large on disk.
Work Queue Size (Number Of Requests)	true	Number of requests in the server's work queue waiting to be processed, averaged over the update interval. There is no resource identifier associated with this gauge. The monitored resource is the server's work queue. 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.

Working with the collect-support-data tool

The PingDirectory 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` archive 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` archive.

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 varies between systems. However, the tool always tries to get the same information across all systems for the target PingDirectory 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

> **Note**
>
> This topic applies only to the PingDirectory server.

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 rerun the specific data collector command and send the results to your authorized support provider.

Data Collector	Description
status	Runs `status -F` to show the full version information of the PingDirectory server (Unix, Windows).
server-state	Runs `server-state` to show the current state of the PingDirectory server process (Unix, Windows).

JDK commands used in the collect-support-data tool

> **Note**
>
> This topic applies only to the PingDirectory server.

Data Collector	Description
jps	Java Virtual Machine Process status tool. Reports information on the JVM (Linux, Windows, Mac OS).
jstack	Java Virtual Machine Stack Trace. Prints the stack traces of threads for the Java process (Linux, Windows, Mac OS).
jstat	Java Virtual Machine Statistics Monitoring Tool. Displays performance statistics for the JVM (Linux, Windows, Mac OS).
jinfo	Displays the Java configuration information for the Java process (Linux, Windows, Mac OS).

Linux commands used in the collect-support-data tool

> **Note**
>
> This topic applies only to the PingDirectory server.

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

> **Note**
>
> This topic applies only to the PingDirectory server.

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.
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

> **Note**
>
> This topic applies only to the PingDirectory server.

You can invoke `collect-support-data` as an administrative task, including as a recurring task that can be automatically invoked on a regular basis.

Learn more in About recurring tasks and task chains.

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 instead of in parallel. This action reduces the initial memory footprint of this tool at the cost of taking longer to complete.
- `--reportCount {count}`: Specifies the number of reports generated for commands that support sampling, such as `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, such as `mpstat`. This option must have a value greater than 0 (zero). If this option is not specified, it defaults 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 aren't collected by default because they impact the performance of a running server.
- `--comment {comment}`: Provides the ability to submit any additional information about the collected data set. The comment is 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. More information: `--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 PingDirectoryProxy server.
- `--securityLevel`: Specifies the degree to which the tool will attempt to obscure or omit potentially sensitive information. A value of `none` indicates that the tool won't attempt to obscure or redact any information. A value of `obscure-secrets` indicates that the tool will attempt to obscure secret information (like the values of sensitive configuration properties) and omit log files containing user data (like the data recovery log). A value of `maximum` indicates that the tool will take even more drastic measures, such as omitting access log files and obscuring attribute values in entry DNs and search filters, but at the risk of omitting information that could be useful in investigating the associated issue. If this is not provided, a value of `obscure-secrets` will be used by default.

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.

   Example:

```
$ bin/collect-support-data --hostname 127.0.0.1 --port 389 \
  --bindDN "cn=Directory Manager" --bindPassword secret
```

3. Email the `.zip` archive to your Authorized Support Provider.

PingDirectory server troubleshooting information

> **Note**
>
> This topic applies only to the PingDirectory server.

The PingDirectory 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. Several messages are written to this file on startup and shutdown, but while the server is running, there is normally little information written to it. However, if a problem occurs, 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, and 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, because the server provides a separate debug logging facility.
- `INFORMATION`: Used for informational messages that can be useful from time to time but aren't 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 doesn't prevent the server from completing the task it was working on. These warnings aren't 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 aren't 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 might leave the server unable to continue processing operations normally.

The messages written to the error log can be filtered based on their severities. The error log publisher has a `default-severity` property, which can 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
```

For the purposes of this configuration property, any underscores in category or severity names should be replaced with dashes. Also, severities aren't inherently hierarchical, so enabling the DEBUG severity for a category won't automatically enable logging at the INFORMATION, MILD_WARNING, or MILD_ERROR severities.

The error log configuration can be altered on the fly using tools like **`dsconfig`**, the admin 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 can be activated with a different set of default severities to debug a short-term problem, and then that logger can be removed after the problem is resolved so that the normal error log doesn't 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 several 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. 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 PingDirectory 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 can be used to perform the operation in a non-interactive manner with the `dsconfig` command. Other information written for each change includes:

- Time 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 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 can 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 can 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 can contain.

The 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 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, and so on.

Setup log

The `setup` command 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 might be used if a file with that name already exists because the `setup` command has already been run. The full path to the setup log file is provided when the `setup` command has completed.

Tool log

Many of the administrative tools provided with the server (for example, `import-ldif`, `export-ldif`, `backup`, `restore`, and so on) 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 using the server tasks interface, log messages generated while the task is running can alternately be written to the server error log file.

je.info and je.config files

The primary datastore used by the PingDirectory server is the Oracle Berkeley DB Java Edition (JE). The PingDirectory 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 can 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 PingDirectory 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 PingDirectory server and the PingDirectoryProxy 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.

Monitor entries

> **Note**
>
> This topic applies only to the PingDirectory server.

While the 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 environment used by the associated backend.
- Disk Space Usage Monitor Entry (cn=Disk Space Usage,cn=monitor) – Provides information 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 server version.
- Work Queue Monitor Entry (cn=Work Queue,cn=monitor) – Provides information about the state of the server's 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.

Server troubleshooting tools

The 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 PingDirectory 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 several 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 PingDirectory server provides an LDIF connection handler that provides a way to request operations that don't 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 in the `config/auto-process-ldif` directory. This directory doesn't exist by default, but if you create it and place an LDIF file in it that contains one or more changes to be processed, the LDIF connection handler applies those changes.

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

> **Note**
>
> This topic applies only to the PingDirectory server.

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 can 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

> **Note**
>
> This topic applies only to the PingDirectory server.

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 PingDirectory 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 admin 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.

  Example:

  ```
  $ 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`.

  Example:

  ```
  $ bin/profile-viewer --fileName logs/profile.20110101000000Z --useGUI
  ```

Oracle Berkeley DB Java Edition utilities

> **Note**
>
> This topic applies only to the PingDirectory server.

The Oracle Berkeley DB Java Edition (JE) itself provides several 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 Server is written entirely in Java, it is possible to use standard Java debugging and instrumentation tools when troubleshooting problems with the PingDirectory server. In many cases, obtaining the full benefit of these tools requires access to the 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 several 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 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 PingDirectory 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 Server), and the frame of the stack trace can 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 PingDirectory 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 can 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 PingDirectory 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 PingDirectory 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.

Garbage collection diagnostic information

> **Note**
>
> This topic applies only to the PingDirectoryProxy server.

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 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
```

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 PingDirectory server. In some cases, problems with the underlying system can be directly responsible for the issues seen with the 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 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 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 server is performing poorly or is unresponsive, there can be a resource constraint elsewhere (for example, waiting on disk or memory access, or excessive lock contention), or the JVM can be performing other tasks like stop-the-world garbage collection that cannot be run heavily in parallel.

Per-CPU utilization

To investigate CPU consumption on a per-CPU basis, use the `mpstat` command with a time interval in seconds, like:

```
mpstat 5
```

On Linux systems, it might be necessary to add "`-P ALL`" to the command, like:

```
mpstat -P ALL 5
```

Among other things, this shows the percentage of time each CPU has spent in user time, system time, and idle time. If the overall CPU utilization is relatively low but `mpstat` reports that one CPU has a much higher utilization than the others, there might be a significant bottleneck within the server or the JVM might be performing certain types of garbage collection which cannot be run in parallel. On the other hand, if CPU utilization is relatively even across all CPUs, there is likely no such bottleneck and the issue might be elsewhere.

Per-process utilization

To investigate CPU consumption on a per-process basis, use a command such as the `top` utility on Linux. If a process other than the Java process used to run the PingDirectory server is consuming a significant amount of available CPU, it might be interfering with the ability of the server to run effectively.

Examining disk utilization

If the underlying system has a very high disk utilization, it can adversely impact 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 can 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
```

Several 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 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 several 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 might 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 server is a network-based application, it can be valuable to observe the network communication that it has with clients. The server itself can provide details about its interaction with clients by enabling debugging for the protocol or data debug categories, but there can be several 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 can require root access.

Common problems and potential solutions

This section describes several different types of problems that can occur and common potential causes for them.

General troubleshooting methodology

When a problem is detected, the following general methodology to isolate the problem are recommended.

1. Run the `bin/status` tool or look at the server status in the admin 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 Data 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 server requires that Java be installed on the system and made available to the server, and it must be installed before 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

> **Note**
>
> This topic applies only to the PingDirectory server.

If the version of the 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 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 server before running the `setup` tool. This protects an existing server installation from being inadvertently updated in a manner that could harm an existing configuration or data set.

If the server has been previously used and if you want to perform a fresh installation, it is recommended that you first remove the existing installation, create a new one and run `setup` in that new installation. However, if you are confident that there is nothing of value in the existing installation (for example, if a previous attempt to run `setup` failed to complete successfully for some reason but it will refuse to run again), the following steps can be used to allow the `setup` program to run:

- Remove the `config/config.ldif` file and replace it with the `config/update/config.ldif.{revision}` file containing the initial configuration.
- If there are any files or subdirectories below the `db` directory, then remove them.
- If a `config/java.properties` file exists, then remove it.
- If a `lib/setup-java-home` script (or `lib\set-java-home.bat` file on Microsoft Windows) exists, then remove it.

The server will not start

If the server does not start, then there are several 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. 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 Diretory Server could not acquire an exclusive lock on file
/ds/the/locks/server.lock: The exclusive lock requested for file
/ds/the/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, 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 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 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 several 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 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 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 server, or reconfigure the server to reduce the amount of memory that it attempts to use.
- The 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 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 might 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 server configuration using an officially-supported tool like `dsconfig` or the admin 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 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 several 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 configuration file, you can look at previous archived 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 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 server installation, so that they are owned by the user or role under which the server should run. For example, if the 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/the
```

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 PingDirectory server was previously running but is no longer active, then the potential reasons include the following:

- The PingDirectory 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 PingDirectory 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 PingDirectory 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 PingDirectory 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 PingDirectory 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 servers will shut down in an out of memory condition, a low disk space error state, or for running out of file descriptors. The 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 won't accept client connections

You can first check the current server state by using the `bin/server-state` command. If the server doesn't appear to be accepting connections from clients, then potential reasons include the following:

- The server isn't running.
- The underlying system on which the server is installed isn't running.
- The server is running but isn't 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 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 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 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 doesn't 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 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 it appears that the problem is with the 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.

- 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 might 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 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 (because of garbage collection). The alert will include details of the pause.
- If the JVM in which the 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 server is running are not receiving TCP acknowledgement.
- If the system on which the 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.

The server is slow to respond to client requests

If the server is running and does respond to clients, but clients take a long time to receive responses, then the problem can be attributable to several potential problems. In these cases, use the Periodic Stats Logger, which is a valuable tool to get per-second monitoring information on the 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 admin console to see the 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 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 several pauses and can include tuning recommendations to help alleviate the problem.

- If the 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 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 PingDirectory 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 PingDirectoryProxy servers or load balancers or both to route requests away from the affected server. As of PingDirectory version 3.1 or later, the PingDirectory server will indicate that it is in a degraded status and the PingDirectoryProxy server will route around it.
- If the PingDirectory 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 because of 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

> **Note**
>
> This topic applies only to the PingDirectory server.

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 might provide information about the nature of the underlying problem. Potential reasons that replication might 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 PingDirectory 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 server.

How to regenerate the server ads-certificate

> **Note**
>
> This topic applies only to the PingDirectory server.

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 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

> **Note**
>
> This topic applies only to the PingDirectory server.

After migrating from a Sun/Oracle configuration to a PingDirectory server, follow the tuning procedures in Configuring the PingDirectory server for Oracle compatibility if the PingDirectory server behaves differently from the Sun/Oracle server.

Troubleshooting ACI evaluation

> **Note**
>
> This topic applies only to the PingDirectory server.

The 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 might 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 because of 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"
```

After the logger has been 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 Candidates

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 admin console

If a problem arises when trying to use the admin console, then potential reasons for the problem can 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 PingDirectory server is online. If it is online, then the access log can provide information about the reasons for the authentication failure.
- If a problem occurs while attempting to interact with the PingDirectoryProxy server instance using the admin console, then the access and error logs for that PingDirectory server instance might provide additional information about the underlying problem.

Problems with the admin console: JVM memory issues

Console runs out of memory (PermGen). An inadequate PermSize setting in the server, while hosting web applications like the admin console can 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.

Troubleshooting global index growing too large

If the global index appears to be growing too large, you can reload from the backend directory servers.

About this task

> **Note**
>
> This topic applies only to the PingDirectoryProxy server.

Steps

- To override the configuration of the `prime-index-source` property, run the `reload-index` tool with the `--fromDS` subcommand.

  Example:

  You can do this on a one-off basis if the global index appears to be growing too large.

  ```
  $ bin/reload-index \
    --bindPassword password \
    --baseDN "dc=example,dc=com" \
    --fromDS
  ```

Recovering forgotten Proxy User password

If you forget your `cn=Proxy User` password, you can recover it using different methods.

About this task

> **Note**
>
> This topic applies only to the PingDirectoryProxy server.

Steps

- To recover your password, do any of the following:

  Choose from:

  - The PingDirectory server supports a feature called password retirement that allows you to set a new password while allowing the current password to remain usable for a specified period of time. This can be used to set a new password for the account while still allowing the current password to remain temporarily available so that existing PingDirectoryProxy server instances will still be able to use the current password to authenticate until you can update them with the new password. To enable password retirement, update the `password-retirement-behavior` property in the root password policy for all PingDirectory server instances to include a value of `retire-on-administrative-reset`, and set the `max-retired-password-age` property to indicate how long you want the current password to remain valid after setting a new password. Then, reset the Proxy User password for all of the PingDirectory server instances, and update all of the PingDirectoryProxy server instances to use the new password in their LDAP external server configuration.
  - If you do not know the clear-text value for the password but you have other PingDirectoryProxy server instances set up to use that password, then you can use the encoded password value from the LDAP external server configuration entry of one of the existing PingDirectoryProxy servers when creating the LDAP external server for new PingDirectoryProxy server instances.

- Create a new root user in the directory server instances with the appropriate set of privileges and have the new PingDirectoryProxy server instance use that account to authenticate. To use the new account, update all of the other PingDirectoryProxy instances.

Problems with the HTTP Connection Handler

> **Note**
>
> This topic applies only to the PingDirectory server.

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 (for example, "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

> **Note**
>
> This topic applies only to the PingDirectory server.

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 might 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.

Troubleshooting the PingDataSync server

There are several ways to troubleshoot issues with the PingDataSync server.

This section contains the following topics:

PingDataSync gauges

The PingDataSync server provides several built-in gauges to monitor PingDataSync performance. These gauges are listed in the following table:

Directory Server Gauges

Gauge Name	Enabled by default?	Description
Available File Descriptors	true	Monitors the number of file descriptors available to the server process. The server allows for an unlimited number of connections by default, but is restricted by the file descriptor limit on the operating system. 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. Running out of available file descriptors can lead to unpredictable behavior and severe system instability.

Gauge Name	Enabled by default?	Description
Certificate Expiration (Days)	true	Monitors the expiration dates of key server certificates. A server certificate expiring can cause server unavailability, degradation, or loss of key server functionality. Certificates nearing the end of their validity should be replaced as soon as possible. See the status tool, or Status in the admin console, for more information about server certificates and how they are managed.
CPU Usage (Percent)	true	Monitors server CPU use and provides an averaged percentage for the interval defined. The monitored resource is the host system's CPU, which does not include a resource identifier. 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.
Destination Unavailable Seconds	true	Gauge that raises an alarm if the Sync Destination is not available.
Disk Busy (Percent)	true	Monitors the percentage of disk use time averaged over the specified update interval. This gauge requires that the Host System Monitor Provider be enabled and that any monitored disks be registered using the disk-devices property of that configuration object. The resource identifier for this gauge is the disk device name. Use the `iostat` command or a similar system utility to see a list of disk device names. A separate gauge monitor entry will be created for each monitored disk.
HTTP Processing (Percent)	true	Monitors the percentage of time that request handler threads spend processing HTTP requests. This percentage represents the inverse of the server's ability to handle new requests without queuing.
JVM Memory Usage (Percent)	true	Monitors the percentage of Java Virtual Machine (JVM) memory that is in use. This value naturally fluctuates because of garbage collection, so the minimum value within an interval is reported since it is a better indication of overall memory growth. When the memory usage exceeds 90%, this should be reported to customer support since the server is either misconfigured or has a memory leak. As memory usage approaches 100%, the server is more and more likely to experience garbage collection pauses, which leave the server unresponsive for a long time. Restarting the server is likely the only remedy for this situation. Before restarting the server, run `collect-support-data` and capture the output of 'jmap -histo ' to provide to customer support. The pid of the server can be found from `/logs/server.pid`.

Gauge Name	Enabled by default?	Description
License Expiration (Days)	true	Monitors the expiration date of the product license. An expired license will cause warnings to appear in the server's logs and in the status tool output. Request a license key through the Ping Identity licensing website https://www.pingidentity.com/en/account/request-license-key.html or contact sales@pingidentity.com. Use the dsconfig tool to update the License configuration's license key property.
Memory Usage (Percent)	false	Monitors the percentage of memory use averaged over the update interval defined. The monitored resource is the host system's memory use, which does not have a resource identifier. Some operating systems, including Linux, use the majority of memory for file system cache, which is freed as applications need it. If memory use is high, check the applications that are running on the server.
Strong Encryption Not Available	true	The JVM does not appear to support strong encryption algorithms, like 256-bit AES. The server will fall back to using weaker algorithms, like 128-bit AES. To enable support for strong encryption, update your JVM to a newer version that supports it by default, or install or enable the unlimited encryption strength jurisdiction policy files in your Java installation.

PingDataSync log files

The following log files are specific to PingDataSync, 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 because of 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 PingDirectory 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 admin 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
```

The status tool

The `status` tool outputs the health of the server. It 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.

The collect-support-data tool

About this task

PingDataSync 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 might 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.

Steps

1. Go to the server root directory.
2. Run the `collect-support-data` tool. Include the host, port number, bind distinguished name (DN), and bind password.

```
$ bin/collect-support-data \
--hostname 100.0.0.1 --port 389 \
--bindDN "cn=Directory Manager"
--bindPassword secret
```

3. Email the `.zip` file to a support provider.

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 be 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.
changeNumber	Specifies the change number from the source server assigned to the modification.

Sync log element	Description
replicationCSN	Specifies the replication change sequence number from the source server. **Note** This only appears if the change on the source server was made through replication.
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.

```
[17/Nov/2021:15:57:39.562 -0600] category=SYNC severity=INFORMATION msgID=1893728293 op=7
changeNumber=59 pipe="ds1_to_PingOne_Destination" msg="Detected MODIFY of
uid=user.0,ou=People,dc=example,dc=com at ldap://localhost:1389"
```

Sync log example 2

The next example shows another synchronization operation. This was triggered by a `MODIFY` operation on the source server and was successfully synchronized to the destination.

```
[17/Nov/2021:16:22:30.096 -0600] category=SYNC severity=INFORMATION msgID=1893728306 op=8
changeNumber=60 pipe="ds1_to_PingOne_Destination" class="ds1 Users Sync Class"
msg="Synchronized ADD of uid=user.2,ou=People,dc=example,dc=com at ldap://localhost:1389
by modifying entry id=32659530-3c8c-4f9d-be77-390293087532 (after MODDN, new RDN: uid=user.2)
at https://api.pingone.com/v1"
```

Sync log example 3

The next example shows a failed synchronization attempt on a `MODIFY` operation from the source server that could not be synchronized on the destination server. The log displays the LDAP Data Interchange Format (LDIF)-formatted modification that failed, where the source server added two additional `mobilePhone` values for an entry. This would cause a schema violation on the destination server because the destination (in this example, PingOne) only allows a single value for `mobilePhone`.

```
[17/Nov/2021:15:56:39.294 -0600] category=SYNC severity=SEVERE_WARNING msgID=1893859389 op=6
changeNumber=58 pipe="ds1_to_PingOne_Destination" class="ds1 Users Sync Class"
msg="Detected MODIFY of uid=user.0,ou=People,dc=example,dc=com at ldap://localhost:1389,
but failed to apply this change because: Attribute 'mobilePhone' has 3 values in the entry so
the value cannot be constructed since a single value is required
(id=1893859350 ResourceOperationFailedException.java:126 9.0.0.0-20211117202405.000Z-0bc8a906).
Details: Source change detail:
dn: uid=user.0,ou=People,dc=example,dc=com
changetype: modify
add: mobile
mobile: +1 111 111 1111
mobile: +1 999 999 9999
-
replace: modifiersName
modifiersName: cn=Directory Manager,cn=Root DNs,cn=config
-
replace: modifyTimestamp
modifyTimestamp: 20211117215425.236Z
-
 Equivalent destination changes:
dn: uid=user.0,ou=People,dc=example,dc=com
changetype: modify
replace: mobilePhone
mobilePhone: +1 164 286 4924
mobilePhone: +1 111 111 1111
mobilePhone: +1 999 999 9999
-
 Full source entry:
dn: uid=user.0,ou=People,dc=example,dc=com
mail: user.0@example.com
initials: AOR
homePhone: +1 295 940 2750
pager: +1 604 109 3407
givenName: Anett
employeeNumber: 0
telephoneNumber: +1 594 307 3495
mobile: +1 164 286 4924
mobile: +1 111 111 1111
mobile: +1 999 999 9999
sn: Rezzik
cn: Anett Rezzik
userPassword: **
description: This is the description for Anett Rezzik.
street: 22411 Birch Street
st: PA
postalAddress: Anett Rezzik$22411 Birch Street$Rhinelander, PA  98160
uid: user.0
l: Rhinelander
postalCode: 98160
objectClass: top
objectClass: person
objectClass: organizationalPerson
objectClass: inetOrgPerson
Mapped destination entry:
```

```
dn: uid=user.0,ou=People,dc=example,dc=com
email: user.0@example.com
mobilePhone: +1 164 286 4924
mobilePhone: +1 111 111 1111
mobilePhone: +1 999 999 9999
name: { "family":"Rezzik", "formatted":"Anett Rezzik", "given":"Anett" }
population: { "name":"Users", "id":"dd24a03b-bde0-463c-ae76-584f5925fe3d" }
primaryPhone: +1 594 307 3495
resourceType: user
username: user.0
"
```

Troubleshooting synchronization failures

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 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 distinguished name (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.

While many PingDataSync 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 caused by 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 might 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 because of 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 before 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 might 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 might 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 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 several 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 several 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 can 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 might 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 admin 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 can 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 won't accept client connections

Check the current server state by using the `bin/server-state` command. If the server doesn't appear to be accepting connections from clients, reasons can include the following:

- The server isn't running.
- The underlying system on which the server is installed isn't running.
- The server is running, but isn't 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 Secure Sockets Layer (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 might 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. 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 doesn't 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 admin console

If a problem occurs when trying to use the admin 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 admin 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 Secure Sockets Layer (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 PingDirectory 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 because of 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 Java Virtual Machine (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, go 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
   ```

FIPS Compliance for PingDirectory

The Federal Risk and Authorization Management Program (FedRAMP) could require that United States government agencies and organizations that work with those agencies ensure that their cloud-based services adhere to either the FIPS 140-2 or FIPS 140-3 specification. These specifications define requirements for cryptographic processing.

Introduction to FIPS compliance

The PingDirectory, PingDirectoryProxy, and PingDataSync servers support running in FIPS-compliant mode using either the FIPS 140-2 or FIPS 140-3 specification.

When setting up in FIPS-compliant mode, the server uses the Bouncy Castle FIPS Java API in approved-only mode, which rejects attempts to use non-FIPS-compliant cryptography.

Differences between FIPS-compliant and non-FIPS-compliant modes

Servers running in FIPS-compliant mode and servers running in non-FIPS-compliant mode differ and have certain incompatibilities.

Some of these are incompatibilities imposed by the specification or the Bouncy Castle FIPS provider. Others allow for better out-of-the-box security in a manner that isn't possible without breaking backward compatibility with existing deployments.

Incompatibilities between FIPS-compliant and non-FIPS-compliant servers

Although the server doesn't have any known substantial cryptographic issues when running in the default non-FIPS-compliant mode, some of the algorithms it uses are either not permitted by the FIPS specification or aren't allowed by the Bouncy Castle FIPS provider.

To preserve backward compatibility with earlier versions, the server still uses those same algorithms when operating in non-FIPS-compliant mode, however it uses a different set of algorithms when set up in FIPS-compliant mode.

> **Note**
>
> Although this shouldn't substantially impact compatibility with clients, it does mean that servers running in FIPS-compliant mode cannot participate in the same topology as servers running in non-FIPS-compliant mode, nor is it possible to directly replicate between servers running in FIPS-compliant mode and servers running in non-FIPS-compliant mode.
> It's also not possible to update or reconfigure an existing server running in non-FIPS-compliant mode to operate in FIPS-compliant mode.

To migrate an existing non-FIPS-compliant topology to one operating in FIPS-compliant mode, you must:

1. Create a second topology in which the servers are set up in FIPS-compliant mode.
2. Migrate the data to the new topology.
3. Start serving client requests from that new topology.

> **Note**
>
> Although you can't replicate data directly between FIPS-compliant and non-FIPS-compliant instances, the PingDataSync server supports synchronizing changes between the topologies.

> **Note**
>
> If you're not using automatic backend server discovery for both types of servers, the PingDirectoryProxy server can forward requests to a mix of FIPS-compliant and non-FIPS-compliant servers. This allows you to migrate from one topology to another in a manner that's transparent to clients.

Changes in supported key store types

The server uses key stores to hold certificates needed for TLS negotiation and authentication. Historically, it supported the Java KeyStore (JKS) and PKCS #12 (a standard format described in RFC 7292) file-based key store types. However, these key store types could use cryptographic algorithms that aren't permitted in FIPS-compliant mode. Instead, use the Bouncy Castle FIPS-compliant Key Store (BCFKS) format. BCFKS is similar to PKCS #12 but relies only on cryptographic algorithms that are allowed by the Bouncy Castle FIPS provider in approved-only mode.

The `manage-certificates` tool has been updated to provide support for BCFKS key store types. For servers operating in non-FIPS-compliant mode, it can interact with any of the JKS, PKCS #12 or BCFKS key store types, and the `copy-keystore` subcommand is used to convert a JKS or PKCS #12 key store to a BCFKS key store.

> **Note**
>
> For servers operating in FIPS-compliant mode, only the BCFKS key store format is allowed.

The server also supports the standard PKCS #11 cryptographic API that can be used to interact with tokens like hardware security modules (HSMs). PKCS #11 tokens can be used in either FIPS-compliant mode or non-FIPS-compliant mode, however you should use a FIPS-compliant HSM when running in FIPS-compliant mode. The `manage-certificates` tool also supports interacting with PKCS #11 tokens.

TLS for network communication

When running in non-FIPS-compliant mode, you can configure the server to accept only encrypted, only unencrypted, or a mix of encrypted and unencrypted network communication.

When setting up the server in FIPS-compliant mode, you must enable TLS encryption. Unencrypted communication won't be enabled. The server can optionally accept connections that are initially unencrypted, but the StartTLS extended request must be used to secure communication on those connections before other types of requests are allowed.

Because secure communication is required, you must configure the server with appropriate certificate key and trust stores during setup. Learn more in Setting up certificate key and trust stores.

Data encryption

Using data encryption to protect data at rest is optional in non-FIPS-compliant mode but is required when setting up the server in FIPS-compliant mode.

You must configure the server with at least one encryption settings definition. Learn more in Setting up data encryption.

> **Note**
>
> When creating an encryption settings definition from a passphrase (whether user-supplied or automatically generated), servers running in non-FIPS-compliant mode default to using a 128-bit AES encryption key. When setting up a server in FIPS-compliant mode, `setup` uses a 256-bit AES encryption key.

Strong administrative passwords

When setting up the server in non-FIPS-compliant mode, you should choose a strong password for the initial root user that conforms to the following requirements:

- The password must be at least 12 characters long.
- The password must not be contained in a dictionary of words from a variety of languages.
- The password must not be contained in a dictionary of commonly used passwords.

In non-FIPS-compliant mode, you can ignore those rules and choose a weaker password through the `--allowWeakRootUserPassword` argument. You can't use this argument when setting up the server in FIPS-compliant mode, and you will be required to provide what the server considers a strong password.

The minimum password length requirement is increased from 12 characters to 14 characters because this is the minimum length required by the Bouncy Castle FIPS-compliant PBKDF2 implementation when running in approved-only mode.

Reduced available password storage schemes in FIPS-compliant mode

More password storage schemes are available in the out-of-the-box configuration when setting up a server in non-FIPS-compliant mode than in FIPS-compliant mode.

The only schemes available in the out-of-the-box configuration for a FIPS-compliant server are:

- AES256
- PBKDF2
- SSHA256
- SSHA384
- SSHA512

The PBKDF2 password storage scheme is the default scheme for root users and topology administrators. The SSHA256 password scheme is the default scheme for non-administrative users.

> **Note**
>
> In FIPS-compliant mode, the server defaults to the PBKDF2WithHmacSHA256 algorithm rather than PBKDF2WithHmacSHA1, which is used by default for backwards compatibility in non-FIPS-compliant mode. The SSHA256, SSHA384, and SSHA512 schemes use a default salt length of 128 bits in FIPS-compliant mode as opposed to the default length of 64 bits in non-FIPS-compliant mode.

The following password storage schemes aren't available in FIPS-compliant mode because they depend on the non-FIPS-compliant Bouncy Castle library, which is not compatible with the FIPS-compliant library:

- ARGON2
- BCRYPT
- SCRYPT

The following schemes are available in the default configuration for non-FIPS-compliant mode for the purpose of backward compatibility and are excluded from the out-of-the-box configuration for a FIPS-compliant server:

- AES
- BASE64
- BLOWFISH
- CLEAR
- CRYPT
- MD5
- SHA
- SMD5
- SSHA

> **Note**
>
> You should avoid using these password storage schemes whenever possible.

Reduced available SASL mechanisms

When running in non-FIPS-compliant mode, the server has some non-recommended Simple Authentication and Security Layer (SASL) mechanisms available in the out-of-the-box configuration for purposes of backward compatibility that aren't included in the FIPS-compliant mode default configuration.

These mechanisms include:

- ANONYMOUS
- CRAM-MD5
- DIGEST-MD5

Setting up the server in FIPS-compliant mode

You can't have both FIPS-compliant and non-FIPS-compliant servers in the same topology, and you can't update an existing non-FIPS-compliant server to run in FIPS-compliant mode.

Because of this, if you want to run a server in FIPS-compliant mode, you must set up a new instance. This can be done in the following two ways:

- Use `setup` with an appropriate set of arguments.
- Use `manage-profile setup` with a profile that includes the necessary setup arguments, along with any other configuration changes, extensions, and other files that you might want included in the installation.

> **Note**
>
> You can run servers with different levels of FIPS compliance in the same topology.

Ensure sufficient entropy

Strong cryptography requires a reliable source of high-quality random data.

On some systems, the OS-provided random number generator, such as `/dev/random` on Linux systems, might block if there's not enough entropy available to keep up with the demand for strong random data, which can severely impede server performance. This is especially likely when running the server in a virtual machine or in a container because it's less likely to have access to the entropy stream from the underlying host system.

When running in non-FIPS-compliant mode, the server can work around this problem by using an alternative random number generator, such as `/dev/urandom` on Linux, that uses cryptographic techniques to ensure that it can still provide a high-quality stream of random data that won't block even when available entropy is exhausted on the underlying system. However, the Bouncy Castle FIPS-compliant random number generator doesn't support this alternative, and it's likely to block for long periods of time if the server is installed in a container or virtual machine.

This isn't a problem that's likely to go unnoticed because the server is likely to appear completely unresponsive for many minutes at a time if the random number generator blocks because of a lack of entropy. It's likely to block for long periods of time, especially if the server is installed in a container or virtual machine.

Resolve entropy exhaustion

To help diagnose the problem, the installer attempts to monitor available system entropy when setting up the server in FIPS-compliant mode and displays a warning message if entropy drops too low. Similarly, if the server is running in FIPS-compliant mode, it continuously monitors available system entropy and logs a warning message and raises an alarm if entropy drops low enough that the server is likely to become unresponsive.

If entropy exhaustion is a problem, the best options to address it include:

- If the server is running in a virtual machine or container, you might be able to configure it with access to the underlying host system's entropy pool if that's not already the case.
- Install a hardware random number generator on the system and ensure that the server can access it even when running in a container or virtual machine.
- Install an entropy-supplementing daemon, such as rngd, to keep the OS-provided random number generator topped off and able to generate high-quality random data without blocking.

Setting up certificate key and trust stores

About this task

Because FIPS-compliant mode requires secure communication, you must provide arguments that indicate how the server should obtain the certificate chain, private key, and trusted certificate information that it should use during TLS negotiation.

Steps

- Configure the server with appropriate key and trust stores during setup. Choose from:

  Choose from:

  - If you have existing key and trust stores in the BCFKS format:
  - Use the `--useBCFKSKeyStore` and `--useBCFKSTrustStore` arguments to provide the paths to those stores.
  - Use either the `--keyStorePassword` or `--keyStorePasswordFile` argument to specify the PIN needed to access the contents of the key store.
  - Use either the `--trustStorePassword` or `--trustStorePasswordFile` argument to specify the PIN needed to access the contents of the trust store.

    > **Note**
    >
    > Unlike the JKS format, a PIN is always required when using a BCFKS key store, even if you don't need to access the private key.

  - If you have existing key and trust stores in a non-BCFKS format:
  - Convert the non-BCFKS files using `manage-certificates copy-keystore` with the `--destination-key-store-type BCFKS` argument.
  - Follow the steps for existing key and trust stores in the BCFKS format.
  - If you have PEM files containing the certificate chain and private key from a certificate authority, and you want to use them to generate new BCFKS key and trust stores:
  - Use the `--certificateChainPEMFile` and `--certificatePrivateKeyPEMFile` arguments to specify the paths to those files.
  - If you have PEM files containing trusted certificates that you want to include in a new BCFKS trust store, you can use the `--trustedCertificatePEMFile` argument to provide the paths to those files.
  - If the listener certificate chain and private key that you want to use reside in a PKCS #11 token:
  - Use the `--usePKCS11KeyStore` argument to enable that support for creating a BCFKS key store.

    > **Important**
    >
    > PingDirectory only supports key store creation using PKCS #11 tokens. In order to create a trust store, you must use either the `--useBCFKSTrustStore` or `--trustedCertificatePEMFile` arguments in conjunction with `--usePKCS11KeyStore`.

  - If the Java virtual machine (JVM) has not been pre-configured with the necessary PKCS #11 provider, then use the `--pkcs11ProviderConfigFile` argument to specify the path to the necessary provider configuration file.
  - Use either the `--keyStorePassword` or `--keyStorePasswordFile` argument to specify the PIN needed to access the token.

- If you want the server to generate a self-signed certificate and use it to create BCFKS key and trust stores, use the `--generateSelfSignedCertificate` argument.

  > **Important**
  >
  > Self-signed certificates are convenient for testing or evaluation purposes, but they aren't trusted by any clients (by default) and shouldn't be used in production environments.

Setting up data encryption

About this task

Setting up the server in FIPS-compliant mode requires that you enable data encryption.

Steps

- Configure the server with at least one encryption settings definition. Choose from:

  Choose from:

  - If you want the server to generate an encryption settings definition from a passphrase that you provide, use the `--encryptDataWithPassphraseFromFile` argument to specify the path to a file containing that passphrase.

    > **Note**
    >
    > If you provide the same passphrase to each instance, they will generate the same encryption settings definition and will encrypt data in the same way. Also, in many cases, if you know the passphrase used to generate an encryption settings definition, you can use that passphrase to decrypt encrypted data even if the encryption settings definition isn't available.

  - If you have one or more encryption settings definitions that have been exported from another instance:

    1. Use the `--encryptDataWithSettingsImportedFromFile` argument to specify the path to that export file.
    2. Provide the `--encryptionSettingsExportPassphraseFile` argument to specify the path to a file containing the passphrase used to protect the contents of that export.

  - If you want the server to generate an encryption settings definition with a randomly generated passphrase, use the `--encryptDataWithRandomPassphrase` argument.

    > **Note**
    >
    > If you use this argument when setting up multiple instances, then each instance will have a different encryption settings definition, and data encrypted on one instance might not be accessible to other instances. However, you can use it when setting up the first instance in a topology and then export the generated definition and use the `--encryptDataWithSettingsImportedFromFile` argument to import it when setting up additional instances.
    > Because the random passphrase the server generated when creating the definition will not be exposed, you can't use it to decrypt data if that encryption settings definition is not available.

Installing the server in FIPS-compliant mode

Install the server in FIPS-compliant mode with TLS negotiation and data encryption.

About this task

Interactive setup doesn't provide an option to enable FIPS-compliant mode, and there are currently no other supported providers that can be used to enable FIPS-compliant mode.

Steps

- Install the server with one of the supported FIPS specifications.

  Choose from:

  - To set up the server for FIPS 140-2 compliance, add `--fips-provider BCFIPS1` to the set of arguments used when running setup in non-interactive mode or to the server profile's `setup-arguments.txt` file when using `manage-profile setup`.

    Example:

    The following example provides a sample command line that demonstrates the process for setting up the server in FIPS 140-2-compliant mode. The server only accepts TLS-encrypted LDAP on port 636 and TLS-encrypted HTTP on port 443, but doesn't allow unencrypted connections from either LDAP or HTTP clients. BCFKS key and trust stores are generated from information provided in PEM files, and an encryption settings definition is generated from a specified passphrase.

    ```
    ./setup \
      --fips-provider BCFIPS1 \
      --no-prompt \
      --acceptLicense \
      --localHostName ds1.example.com \
      --ldapsPort 636 \
      --httpsPort 443 \
      --baseDN "dc=example,dc=com" \
      --rootUserDN "cn=Directory Manager" \
      --rootUserPasswordFile /path/to/root-pw.txt \
      --maxHeapSize 2g \
      --primeDB \
      --sampleData 10001 \
      --certificateChainPEMFile /path/to/server-cert.pem \
      --certificateChainPEMFile /path/to/ca-cert.pem \
      --certificatePrivateKeyPEMFile /path/to/server-key.pem \
      --trustedCertificatePEMFile /path/to/ca-cert.pem \
      --encryptDataWithPassphraseFromFile /path/to/encryption-passphrase.txt \
      --instanceName ds1 \
      --location example-location \
      --noPropertiesFile
    ```

  - To set up the server for FIPS 140-3 compliance, add `--fips-provider BCFIPS2` to the set of arguments used when running setup in non-interactive mode or to the server profile's `setup-arguments.txt` file when using `manage-profile setup`.

Checking the level of FIPS compliance

You can check a server's level of FIPS compliance by reviewing the properties of its `cn=Version,cn=monitor` entry.

About this task

The following table details the properties to review within the monitor entry:

Property	Values
`fips-compliant-mode`	• `true` indicates that the server is installed in FIPS-compliant mode (using any supported specification). • `false` indicates that the server isn't installed in FIPS-compliant mode.
`fips-140-2-compliant-mode`	• `true` indicates that the server is configured with the FIPS 140-2 specification. • `false` indicates that the server isn't configured with the FIPS 140-2 specification.
`fips-140-3-compliant-mode`	• `true` indicates that the server is configured with the FIPS 140-3 specification. • `false` indicates that the server isn't configured with the FIPS 140-3 specification.

Note

A FIPS-compliant server returns a value of `true` for the `fips-compliant-mode` property and one of the FIPS specification properties. It returns `false` for all other FIPS specification properties.

Steps

- Run `ldapsearch` to return the `cn=Version,cn=monitor` entry.

  Example:

  The following example includes an `ldapsearch` and the resulting entry, which indicates that the server is installed in FIPS 140-2-compliant mode:

```
$ bin/ldapsearch \
  --hostname <hostname> --port <port> --useSSL --trustAll \
  --bindDN "cn=Directory Manager" --bindPasswordFile <password_file> \
  --baseDN cn=Version,cn=monitor "(&)" fips-compliant-mode fips-140-2-compliant-mode fips-140-3-
compliant-mode


  dn: cn=Version,cn=monitor
  fips-compliant-mode: true
  fips-140-2-compliant-mode: true
  fips-140-3-compliant-mode: false

  # Result Code:  0 (success)
  # Number of Entries Returned:  1
```

Changing the level of FIPS compliance

You can change the level of FIPS compliance for an existing FIPS-compliant server.

About this task

You can update a server's FIPS compliance level to any supported FIPS specification. For example, your organization might be required by regulation to update an existing topology of FIPS 140-2-compliant servers to FIPS 140-3 compliance. Alternatively, you might need to transition a FIPS 140-3-compliant server to the FIPS 140-2 specification for technical reasons.

Before you begin

To update the FIPS specification on a FIPS-compliant server, you need to know the following:

- The server's current FIPS specification
- The server's installation method:
  - Using `manage-profile setup`
  - Using `setup`
- The encoded value of the target FIPS specfication:
  - `BCFIPS1` for FIPS 140-2
  - `BCFIPS2` for FIPS 140-3

Steps

- Follow the appropriate steps based upon the type of server installation method:

  Choose from:

  - For servers installed using `manage-profile setup`, run `manage-profile replace-profile`. Supply the `--fips-provider` argument using the encoded value of the target FIPS specification. Learn more about the `manage-profile` tool.

- For servers installed using `setup`:

  1. In `config/java.properties`, edit the value of `-Dcom.unboundid.crypto.FIPS_PROVIDER` to the encoded value of the target FIPS specification.
  2. Run `bin/dsjavaproperties`.
  3. Restart the server.

Troubleshooting:

If the server doesn't operate as expected after the update, revert the server to the previous FIPS compliance level.

PingDirectory Security Guide

The PingDirectory Security Guide provides concepts and procedures to secure and manage the PingDirectory platform.

Introduction

This guide is intended for administrators responsible for installing and managing servers in an enterprise or consumer-facing identity environment. 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.

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 several 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 before 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 Server Administration Guide makes several 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 several 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.

Argument	Description
`--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.
`--encryptDataWithPreExistingEncryptionSettingsDatabase`	Indicates that the server should enable data encryption using the definitions from a pre-existing encryption settings database. This database can be protected with any cipher stream provider supported by the server, configured with data encryption restrictions, and frozen so that its contents are immutable. If you set up the server with a pre-existing encryption settings database, you should use the `manage-profile setup` tool. The server profile must meet the following requirements: • The `setup-arguments.txt` file must include the `--encryptDataWithPreExistingEncryptionSettingsDatabase` argument. • The server profile must contain the `server-root/pre-setup/config/encryption-settings/encryption-settings-db` file, which represents the encryption settings database to use for the new server instance. • The `pre-setup-dsconfig` directory must exist and it must contain one or more `dsconfig` batch files with the changes needed to set up and enable the cipher stream provider to use with the encryption settings database. • The `server-root/pre-setup` directory should include any metadata files that the cipher stream provider needs to access the encryption settings database.

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.

Example

```
$ 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.

Argument	Description
`--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-factory-iteration-count <count>`	Specifies the PBKDF2iteration count to use in deriving the encryption key for the encryption settings definition. You should use this argument when creating a new encryption settings definition. The current OWASP recommendation is to use a PBKDF2 iteration count value of at least 600,000.
`--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.
`--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-factory-iteration-count 60000 \
     --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.
`--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.
`--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.

Argument	Description
`--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

The `encryption-settings import` command with the `import` subcommand provides you with a mechanism to import definitions into the encryption settings database.

About this task
Steps

- To import definitions into the encryption settings database, use the `encryption-settings` command with the `import` subcommand.

  The `import` subcommand supports the following arguments.

Argument	Description
`--input-file` (required)	Specifies the path to the file containing the exported encryption settings definitions.
`--pin-file` (optional)	Specifies the path to a file containing the passphrase used to protect the contents of the encryption settings definitions. If this is not provided, the tool interactively prompts for the passphrase.
`--set-preferred` (optional)	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.

> **Note**
>
> 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.

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/{pingdir}/exported-definitions.esd.
Successfully imported encryption settings definition
CA8A76C13DD5CC3F85A437119D9DC0867396910F64E228962A30FF80B36C3B63 from file
/ds/{pingdir}/exported-definitions.esd.
```

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.

Configuring data encryption restrictions

The PingDirectory server supports several data encryption restrictions that make it harder for unauthorized individuals to access data in an unencrypted form.

About this task

> **Note**
>
> By default, none of the available data encryption restrictions are active in the server.

Steps

- To configure data encryption restrictions, use the `encryption-settings set-data-encryption-restrictions` command with one of the following arguments.

Arguments	Description
`--add-restriction <restriction-name>`	Activates the specified encryption restriction in the server. You can provide this argument multiple times with a single command to add multiple restrictions.
`--remove-restriction <restriction-name>`	Removes the specified encryption restriction from the server. You can provide this argument multiple times with a single command to remove multiple restrictions.
`--remove-all-restrictions`	Removes any data encryption restrictions that are currently in place.
`--add-all-restrictions`	Activates all supported data encryption restrictions that are not already active.

Example:

```
$ bin/encryption-settings set-data-encryption-restrictions \
    --add-all-restrictions
```

After the successful completion of the previous command, you receive a message like the following:

```
Successfully updated the set of active data encryption restrictions.

The updated set of active data encryption restrictions is:
* prevent-disabling-data-encryption.
* prevent-changing-cipher-stream-provider.
* prevent-encryption-settings-export.
* prevent-unencrypted-ldif-export.
* prevent-passphrase-encrypted-ldif-export.
* prevent-unencrypted-backup.
* prevent-passphrase-encrypted-backup.
* prevent-decrypt-file.
```

- To determine which data encryption restrictions are active in the server, use the `encryption-settings get-data-encryption-restrictions` command.

> **Note**
>
> If you are defining data encryption restrictions in the server, freeze the encryption settings database so that these restrictions cannot be modified by anyone without the appropriate passphrase. For more information, see Freezing the encryption settings database.

Freezing the encryption settings database

You can freeze the encryption settings database with a specified passphrase. While it is frozen, the database operates in read-only mode.

About this task

If the encryption settings database is frozen, the server can use the database for data encryption processing but will not allow any of the following:

- Creating new encryption settings definitions
- Importing encryption settings definitions from an exported set
- Removing encryption settings definitions
- Specifying which definition is preferred for new encryption operations
- Adding or removing data encryption restrictions

To make changes to a frozen database, you must unfreeze it by providing the passphrase originally used to freeze it.

Steps

- To freeze the encryption settings database, use the `encryption-settings freeze` command.

  This command supports the `--passphrase-file <path>` argument, which specifies the path to a file containing the passphrase to use for freezing the encryption settings database. If the argument is not provided, the `encryption-settings freeze` command prompts the user for the passphrase.

  Example:

```
$ bin/encryption-settings freeze
Enter the passphrase to use to freeze the encryption settings database:
Confirm the freeze passphrase:
Successfully froze the encryption settings database.
```

- To unfreeze the encryption settings database, use the `encryption-settings unfreeze` command and provide the passphrase originally used to freeze the database.

  This command supports the `--passphrase-file <path>` argument, which specifies the path to a file containing the passphrase to use for unfreezing the encryption settings database. If this argument is not provided, the `encryption-settings unfreeze` command prompts the user for the passphrase.

  Example:

  ```
  $ bin/encryption-settings unfreeze
  Enter the passphrase used to freeze the encryption settings database:
  Successfully unfroze the encryption settings database.
  ```

- To determine whether the encryption settings database is currently frozen, use the `encryption-settings is-frozen` command.

  This command does not require any arguments.

  Example:

  ```
  $ bin/encryption-settings is-frozen
  The encryption settings database was frozen at Mon Mar 06 22:42:10 UTC 2023.
  ```

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.

Important

Before removing a compromised encryption settings definition, you should run the **`encrypt-file --find-encrypted-files`** command to search for encrypted files on the server. If any files are encrypted with a key tied to the compromised encryption settings definition, those files will no longer be accessible after you remove the definition, potentially preventing the server from starting or from functioning properly. If any encrypted files are found, run **`encrypt-file --re-encrypt`** to re-encrypt the files with a different definition before removing the compromised definition.

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.

Property	Description
`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.
`encrypt-ldif-exports-by-default`	Indicates whether any new LDIF exports that are created should 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 and writes it to a file. The server then uses a file-based cipher stream provider to read the passphrase and generate a key for encrypting the contents of the encryption settings database. However, the server supports additional cipher stream providers that 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.
- Use a key generated from a passphrase stored in the Amazon Secrets Manager service.
- Use a key generated from a passphrase stored in the Azure Key Vault service.
- Use a key generated from a passphrase stored in a CyberArk Conjur instance.
- Use a key generated from a certificate stored in a PKCS #11 token.

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 create an Amazon AWS external server configuration definition that provides information needed to interact with the AWS service, including which region to use, the type of authentication to use, and whether to access AWS through an HTTP proxy server:

```
$ bin/dsconfig create-external-server \
  --server-name AWS \
  --type amazon-aws \
  --set authentication-method:access-key \
  --set aws-access-key-id:[KMS_ACCESS_KEY_ID] \
  --set aws-secret-access-key:[KMS_SECRET_ACCESS_KEY] \
  --set region-name:us-east-1
```

Then, create a cipher stream provider with the configured external server and specify which key to use to protect the encryption settings database:

```
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]"
```

Finally, update the global configuration to use the new cipher stream provider. This should be done with the server online to ensure that the existing encryption settings database is re-encrypted with the 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 is not 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 found in the following table.

> **Note**
>
> To sanitize log content as it's being written, see Log sanitization.

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.
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.

Argument	Description
`--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.

Argument	Description
`--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.
`--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.

Argument	Description
`--decompress-input`	Indicates that the input data is gzip-compressed. When operating in decrypt mode, the data is decompressed after it has been decrypted.
`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 supported.

UDP-based communication is in the clear, so a network observer can see all of the log messages. You should only use syslog to log to a local syslog server and have it forward messages to a remote server in a secure manner. TLS encryption for TCP-based communication is optionally supported, so you can safely configure the server to log directly to a remote syslog server.

UDP does not provide any feedback about whether 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.

Extension	Description
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".
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.
`display-certificate-file`	Displays the contents of a file containing one or more PEM-encoded or DER-encoded X.509 certificates.

Subcommand	Description
`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 can 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:c5:71:30: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 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:c5:71:30: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:  -207531030019209390598003353674157617387647003537725397654050699787263240396 4
Elliptic Curve Y-Coordinate:  670793565039084272923789184408894120026594857316835707373651279535545085537 3
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

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:f8:51:83:f2:42
```

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/{pingdir}/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:  20862853790475796319788947166709823976229663879966243650207011227930243221133
Elliptic Curve Y-Coordinate:  47969773922664499050574346494178826942092250865477716840891990625413960212095
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: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.
```

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: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 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:40:08:3a:ff
```

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'.
```

After 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

In production environments, you should use listener certificates issued by a certification authority, not self-signed certificates. Certification authorities typically use restricted life spans for the certificates that they sign, and you might need to replace those certificates on a 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, PEM, or DER file 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 are listed in the following table.

Argument	Description
`--source-certificate-file` *<path>*``	The path to the PEM or DER file that contains the new certificate chain. Either this argument or `--source-key-store-file <path>` must be provided. If you are providing a certificate chain with multiple certificates, order the chain with the server certificate first. Each subsequent certificate should be the issuer for the previous certificate. You can provide `--source-certificate-file` multiple times if the certificates in the chain are in separate files rather than one file.
`--source-private-key-file` *<path>*``	The path to the PEM or DER file that contains the private key that correlates to the new PEM or DER server certificate. This argument must be provided when including `--source-certificate-file <path>`.
`--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. Either this argument or `--source-certificate-file <path>` must be provided.
`--source-key-store-password <password>`	The clear-text password needed to access the contents of the source key store.
`--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, PEM, or DER file. 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.

Argument	Description
`--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, PEM, or DER file. 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, though these providers remain separate and distinct.
`--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.
`--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.

Argument	Description
`--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 illustrates what you see when you run the `replace-certificate replace-listener-certificate` command with the `--help` argument:

```
replace-certificate replace-listener-certificate \
     --bindDN uid=admin,dc=example,dc=com \
     --bindPasswordFile admin-password.txt \
     --source-key-store-file new-listener-certificate-keystore.jks \
     --source-key-store-type JKS \
     --source-key-store-password-file new-listener-certificate-keystore.pin \
     --source-certificate-alias new-listener-cert \
     --key-manager-provider JKS \
     --trust-manager-provider JKS \
     --target-certificate-alias server-cert
```

The following example illustrates replacing a PEM-encoded listener certificate:

```
replace-certificate replace-listener-certificate \
     --bindDN cn=Directory Manager \
     --bindPasswordFile admin-password.txt \
     --source-certificate-file new-listener-certificate-chain.pem \
     --source-private-key-file new-private-key.pem \
     --source-private-key-password-file encrypted-private-key-password.txt \
     --key-manager-provider JKS \
     --trust-manager-provider JKS \
     --target-certificate-alias server-cert
```

The following example demonstrates using the `replace-certificate` 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
bHOHEP6AAAAAAAAArOgjH+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
```

For help fixing replication issues related to broken certificate trust in a topology, see Repairing broken listener certificate trust in replication.

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 between 2048 bits and 3072 bits. Certificates with an elliptic curve key pair are not allowed, nor are certificates with an RSA key smaller than 2048 bits or larger than 3072 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 illustrates what you see when you run `replace-certificate replace-inter-server-certificate` with the `--help` argument:

```
replace-certificate replace-inter-server-certificate \
     --bindDN uid=admin,dc=example,dc=com \
     --bindPasswordFile admin-password.txt \
     --source-key-store-file new-inter-server-certificate-keystore.jks \
     --source-key-store-type JKS \
     --source-key-store-password-file new-inter-server-certificate-keystore.pin \
     --source-certificate-alias new-inter-server-cert
```

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
```

PKCS #11 support in the PingDirectory server

PKCS #11 is a standard that defines an API (also known as cryptoki) for interacting with cryptographic tokens.

They can interact with things including:

- Hardware security modules (HSMs)
- Smart cards
- Cryptographic accelerators

> **Note**
>
> Cryptographic accelerators aren't used as much now because most CPUs include their own native support.

Many PKCS #11 tokens provide the ability to store certificates, and the PingDirectory server supports using these types of PKCS #11 tokens as an alternative to a Java KeyStore (JKS) or PKCS #12 file-based key store. While the PKCS #11 tokens used in production environments are typically associated with hardware devices, it's possible to emulate a PKCS #11 token in software.

Using PKCS #11 in the PingDirectory server

There are two ways to configure the PingDirectory server to use a PKCS #11 token to access the listener certificate.

You can either:

- Set up the PingDirectory server with PKCS #11 support enabled during setup.
- Enable PKCS #11 support after setup.

Regardless of the method, you must perform some initial preparation.

Preparing for PCKS #11 support in the PingDirectory server

Steps

1. Extract the PingDirectory server `.zip` file onto the system.

   In this example, it's assumed that it's in `/demo/PingDirectory`.

2. Make `/demo/PingDirectory` your current working directory.

3. Create a provider configuration file that tells Java how to interact with the PKCS #11 token.

   > **Note**
   >
   > If you use a different type of PKCS #11 token, like an actual hardware security module (HSM), then you must specify the appropriate path to its driver library.

   Consult the documentation for the token that you're using for details, but the provider configuration file will generally look something like the following:

   Example:

   ```
   name = pkcs11
   library = /path/to/provider/library.so
   slotListIndex = 0
   ```

4. Create a file with the user PIN needed to access the token.
5. Ensure that the PKCS #11 token has an appropriate certificate chain to present to clients during TLS negotiation.

   In many cases, you can use the `manage-certificates` tool to accomplish this. When interacting with a PKCS #11 token with `manage-certificates`, you must use the `--keystore` argument to specify the path to the provider configuration file, the `--keystore-type` argument with a value of `PKCS11`, and one of the `--keystore-password`, `--keystore-password-file`, or `--prompt-for-keystore-password` arguments to supply the user PIN.

   Example:

   ```
   $ bin/manage-certificates generate-certificate-signing-request \
        --keystore /path/to/provider.conf \
        --keystore-password-file /path/to/pkcs11/user.pin \
        --keystore-type PKCS11 \
        --alias server-cert \
        --subject-dn "CN=demo.example.com,O=Example Corp,C=US" \
        --key-algorithm EC \
        --key-size-bits 256 \
        --signature-algorithm SHA256withECDSA \
        --subject-alternative-name-dns demo.example.com \
        --subject-alternative-name-ip-address 1.2.3.4 \
        --extended-key-usage server-auth \
        --extended-key-usage client-auth \
        --output-file server-cert.csr \
        --output-format PEM
   ```

   After you have the signed certificate, use the `manage-certificates import-certificate` command to import the certificate chain (the signed certificate, any intermediate authority certificates, and the root authority certificate).

Enabling PKCS #11 support during setup

About this task

If you know that you're going to be using a PKCS #11, you can enable PKCS #11 support when running `setup` or `manage-profile setup`.

Steps

- Run `setup` or `manage-profile setup` with the following important arguments:

  `--usePKCS11KeyStore`

  Indicates that you want to configure the server to use a PKCS #11 token to access the listener certificate.

  `--pkcs11ProviderConfigFile <path>`

  Specifies the path to the provider configuration file that tells the JVM how to access the PKCS #11 token.

  `--keyStorePasswordFile <path>`

  Specifies the path to the file containing the user PIN needed to interact with the PKCS #11 token.

  Example:

```
$ ./setup \
     --no-prompt \
     --noPropertiesFile \
     --acceptLicense \
     --localHostName demo.example.com \
     --ldapPort 1389 \
     --ldapsPort 1636 \
     --httpsPort 1443 \
     --usePKCS11KeyStore \
     --pkcs11ProviderConfigFile config/path/to/provider.conf \
     --keyStorePasswordFile /path/to/pkcs11/user.pin \
     --encryptDataWithPassphraseFromFile config/encryption-settings.pin \
     --baseDN dc=example,dc=com \
     --rootUserDN "cn=Directory Manager" \
     --rootUserPasswordFile config/pre-encoded-root-user-password.txt \
     --instanceName demo-instance \
     --location demo-location

Ping Identity Directory Server 8.3.0.0

Initializing ..... Done
Configuring Directory Server .....

Server tools will be configured with a minimal heap size due to limited system memory
available. If out of memory errors occur, it will be necessary to increase tool memory
settings in java.properties and run dsjavaproperties for the changes to take effect.

Configuring Directory Server ..... Done
Configuring Certificates ..... Done
Starting Directory Server ..... Done

Access product documentation from docs/index.html
```

Note

If you don't specify any trust store-related properties, then `setup` automatically generates a trust store populated with just the listener certificate, which works if the token is configured with a self-signed certificate, or if you're using a certificate signed by an authority that is already included in the JVM's default trust store.

However, if you're using a certificate signed by a private authority, then you likely want to either provide an existing trust store, such as in JKS or PKCS #12 format, or you want to use the `--trustedCertificatePEMFile` argument to specify the paths to PEM files for any appropriate issuer certificates that you want to include in the trust store.

Enabling PKCS #11 support after setup

Before you begin

- Make sure that the token already includes a suitable certificate and that the PKCS #11 provider configuration files and user PIN files exist as described in Preparing for PCKS #11 support in the PingDirectory server.
- Make sure that the trust store has the appropriate trust information for the certificate in the PKCS #11 token. If that certificate is signed by an authority in the Java virtual machine (JVM)'s default set of trusted issuers, or if it's signed by the same private internal authority as the certificate in the current file-based key store, then you can skip this.

But if the certificate in the PKCS #11 token is self-signed, or if it's signed by an authority that the server isn't currently configured to trust, then you must update the trust store with the necessary certificates.

About this task

If you already have an existing PingDirectory server set up with some other type of key store, you can update it to use a PKCS #11 token without needing to set up a whole new instance.

Steps

- Update the server to use the PKCS #11 token instead of the file-based key store.

  1. Enable the PKCS11 key manager provider and give it the appropriate provider configuration file and user PIN file.
  2. Update any appropriate connection handlers to use the PKCS11 key manager provider.

     Example:

     The following batch file demonstrates the configuration changes that you can use to accomplish this.

```
dsconfig set-key-manager-provider-prop \
     --provider-name PKCS11 \
     --set enabled:true \
     --set pkcs11-provider-configuration-file:config/path/to/provider.conf \
     --set key-store-pin-file:config/path/to/pkcs11/user.pin

dsconfig set-connection-handler-prop \
     --handler-name "LDAPS Connection Handler" \
     --set key-manager-provider:PKCS11

dsconfig set-connection-handler-prop \
     --handler-name "LDAP Connection Handler" \
     --set key-manager-provider:PKCS11

dsconfig set-connection-handler-prop \
     --handler-name "HTTPS Connection Handler" \
     --set key-manager-provider:PKCS11

dsconfig set-connection-handler-prop \
     --handler-name "JMX Connection Handler" \
     --set key-manager-provider:PKCS11
```

Note

If you change the key type that the certificate uses, for example, from an RSA key pair to an elliptic curve key pair, you might need to restart the server or disable and re-enable the connection handler.
If you don't do this, attempts to establish new secure connections could fail during TLS negotiation, and the server-side error might indicate that it can't handle the new key type.

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.
`--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 Java KeyStore (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 (for example, a hardware security module) to obtain the certificate chain to be presented to clients using LDAPS, HTTPS, and the StartTLS extended operation. The Java virtual machine (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.

Argument	Description
`--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.
`--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, 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. In most cases, you can update the appropriate key manager provider to reference the key store that you plan to use, as shown in the following 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, as shown in the following 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.

Caching key and trust managers

When you create key and trust manager providers, caching is enabled by default, allowing the manager providers to avoid loading key store and trust store files from disk when establishing connections to process requests.

Invalidating the cache

The manager provider reloads files from the configured key store or trust store and refreshes the cache under any of the following conditions:

- The cached manager for the configured store has a `null` value.
- The path to the cached store doesn't match the path of the configured store.
- The length of the cached store doesn't match the length of the configured store.
- The last-updated time for the cached store doesn't match the last-updated time for the configured store.

Enabling or disabling caching (optional)

You can define whether caching is enabled by using the `enable-key-manager-caching` or `enable-trust-manager-caching` properties. Supply a value of `false` to disable caching, causing manager providers to load managers for each connection. Supply a value of `true` to re-enable caching.

To create a key manager provider with caching disabled, supply the `enable-key-manager-caching` property with a value of `false`, as shown in the following example:

```
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 \
    --set enable-key-manager-caching:false
```

To create a trust manager provider with caching disabled, supply the `enable-trust-manager-caching` property with a value of `false`, as shown in the following example:

```
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 enable-trust-manager-caching:false
```

To re-enable caching, set a value of `true` for the same caching property used to create the manager provider, as shown in the following example:

```
dsconfig set-trust-manager-provider-prop \
    --provider-name JKS \
    --set enable-trust-manager-caching:true
```

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 PingDirectoryProxy 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 1.2
- 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 can 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.

Argument	Description
`--keyStorePasswordFile <path>`	The path to a file containing the password needed to access the key store.
`--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 PingDirectory 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 PingDirectory 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:f7:60:81:48:41

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 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 PingDirectory 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 several 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.

Attribute	Description
`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.
`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.

Attribute	Description
`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.
`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.

Attribute	Description
`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 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.

After 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).
`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.

Property	Description
`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.
`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.

Property	Description
`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.
`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.
`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`
`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.

Property	Description
`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.
`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 can 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.
`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`

Property	Description
`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 can 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 can 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.
`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`.

Property	Description
`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`.
`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 might 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.

Property	Description
`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.
`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.
`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.

Property	Description
`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.
`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.

Property	Description
`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 on whether the assurance constraints were satisfied. This is the default value for the property.

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.

Property	Description
`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.
`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.

Property	Description
`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.
`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.
`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 can 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 several 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 Lightweight Directory Access Protocol (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, the 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 several 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 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 the PingDirectory 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, the 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 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 policy tips to improve performance

Consider these best practices for server performance when configuring your password policies.

For your environment, you might configure password policies for different account types and operations so that your policies apply beyond user password changes. For example, when a user signs on, these policies could be triggered to update the `last-login-timestamp` or validate a password during a bind operation.

Given that password policies can affect many server operations, they can also affect server performance. Poorly configured password policies can lead to some of the following issues:

- Slow password binds, replication, or replication topology changes
- A growing replication backlog
- Timing out of replication topology changes
- Excessive use of heap memory indicated by unexpected garbage collection cycles

> **Tip**
>
> Create explicit password policies scoped to groups of users or organizations and be careful when changing the default password policy.

The following table describes password policy configurations that can limit server performance and corresponding recommendations to avoid these limitations:

Limiting configuration	Recommended configuration
Creating a `last-login-timestamp` that includes milliseconds. This can limit performance as follows: • Every bind triggers an update of the entry. ◦ The updated entry has to be replicated. ◦ The change to the entry gets recorded in the `ds-sync-hist` operational attribute. • If a service account is within the scope of the policy (for example, the default policy or a group for a virtual attribute), and the service account binds frequently, replication or memory garbage collection could affect service availability.	• Limit the resolution of any `last-login-timestamp` attributes using the `yyyymmdd` format. ◦ If you need more precise time resolution, limit the bytes of `replication-history-limit` or the duration of `sync-hist-purge-delay`. • Create special password policies for service accounts and keep updates to a minimum.
Creating a bind scoped validator that accesses an external service (such as a pwned password validator). Every bind validation is as slow as the response of the external service. While this configuration might not have a large impact in many cases, it can limit performance where bind password validation frequency is high. The server performs bind password validation according to the `minimum-bind-password-validation-frequency` configuration property, which is set to 30 days by default. Learn more in Configuring password validators for binds.	Avoid external validation services if you set a high frequency for bind password validation.
Creating a password policy for a large number of users that uses a key derivation function. This can limit performance by significantly increasing the CPU capacity required to process binds. **Tip** While the processing cost for first-time user binds that use expensive password storage schemes is high, subsequent bind attempts by users who have previously authenticated since the server started could be significantly reduced by Encoded password caching.	Consider reducing widespread use of key derivation function password storage schemes.
Creating a password policy that relies on interacting with external services such as HashiCorp Vault, CyberArk Conjur, AWS Secrets Manager, or Azure Key Vault. This can limit performance because each bind authentication is as slow as the response of the external service and can't be mitigated by encoded password caching.	Limit password polices that require external services to sensitive accounts.

Limiting configuration	Recommended configuration
Creating a password policy that updates root user or topology administrator entries, which are stored in the `config.ldif` file. This can limit performance because every bind operation triggers the following: • The `config.ldif` file gets completely rewritten. • The `config.ldif` file gets backed up. ◦ Backups can slow down the completion of `dsreplication` subcommands including `enable`, `disable`, and `remove-defunct-server`. • The server raises an alert. Changes to topology administrator entries are mirrored to other servers in the topology, further affecting performance.	Don't add policy constraints that update the entry for root user or topology administrator accounts.

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

The 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. The 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.

The PingDirectory server 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.The 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.

> **Important**
>
> The PBKDF2 storage scheme is inherently slow. Avoid using this scheme for service accounts unless you have evaluated the computational processing costs involved.

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.

The 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 the PingDirectory server uses for passwords encoded with these schemes should be compatible with the format used by other servers that support them.

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

The 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

The 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 the 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.

Encoded password caching

By default, the server can cache passwords encoded with PBKDF2, bcrypt, scrypt, or Argon2 to help improve authentication performance for passwords encoded with those schemes.

How encoded password caching works

The PingDirectory, PingDirectoryProxy, and PingDataSync servers support several computationally expensive password storage schemes. The encoded password cache can speed up repeated authentication attempts for users with these schemes, as follows:

1. The first time that a user authenticates, the server verifies the provided plaintext password using the expensive storage scheme.
2. The server then caches the encoded representation of the password in memory, along with an alternative salted SHA-256-encoded representation.
3. For all following authentication attempts as the same user, if the server finds their encoded password in the cache, it can attempt to verify their provided plaintext password using the faster SHA-256 algorithm.
4. The server automatically clears the cache whenever you change the configuration for the associated password storage scheme.

This cache can significantly reduce the performance impact of attempting to authenticate multiple connections as the same user in a short period of time, as could happen when a client creates a connection pool. This cost savings might be seen when the PingDirectoryProxy server makes its connections to the PingDirectory server because the Proxy User account has a PBKDF2-encoded password by default, as do the accounts for other root users and topology administrators.

Configuring the cache

The encoded password cache is enabled by default for expensive password storage schemes. The server maintains a separate cache with a default size of 10,000 passwords for each of these schemes.

You can change the size of the cache or disable it entirely by setting the value of the `encoded-password-cache-size` property for the applicable password storage schemes.

For example, the following command sets the cache size to 2000 passwords:

```
$ bin/dsconfig set-password-storage-scheme-prop \
    --scheme-name <password_storage_scheme_name> \
    --set encoded-password-cache-size:2000
```

> **Important**
>
> To disable the cache for a given password storage scheme, set the value of `encoded-password-cache-size` to 0.

Security considerations

The following security considerations apply to encoded password caching:

- The cache doesn't hold the plaintext representations of passwords. All cached passwords are stored in both their original encodings and as a cryptographically secure SHA-256 digest generated from a combination of the provided password and a securely generated 128-bit salt.
- The cache isn't persisted. It's only held in memory. Passwords that are persisted are encoded only with their configured storage schemes.

- The cache only holds encoded passwords. It doesn't include any other information that might be able to associate an encoded password with a user account.
- The cache can't become stale in any way that would affect the accuracy of authentication processing. For example, if a user changes their password, the cache won't allow them to continue authenticating with a former password. It also won't allow a user to continue authenticating after their account has become locked or disabled, or after their password has expired.

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 can 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 can 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 might 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 might 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.

Configuring 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 might be better to only periodically invoke validators to reduce the potential impact on performance. This is especially true for cases in which validation might 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:

- f`orce-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.
- r`eject-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. After 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

The server supports maintaining a record of the most recent successful attempt to sign on.

This isn't 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 isn't 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 shouldn't 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 doesn't 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 isn't 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).

Refer to 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.

Self password changes requiring current passwords

If you set the `password-change-requires-current-password` property to `true`, users must provide their current password when choosing a new password.

You can make these password changes using either a regular LDAP modify operation or a password modify extended operation.

> **Important**
>
> For either method:
>
> - If the user doesn't provide the correct current password, the server rejects the password modify request.
> - Password submissions must be in plain text, not encoded.
>
> Refer to 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.

LDAP modify operation

For a regular LDAP modify operation, the password change request must include modifications to delete the user's current password and add their new password.

The following example uses the `ldapmodify` tool to change a user's password:

```
$ bin/ldapmodify --hostname server.example.com --port 636 --useSSL \
--bindDN "cn=admin,dc=example,dc=com" --bindPassword <bindPassword>
dn: uid=jdoe,ou=People,dc=example,dc=com
changetype: modify
delete: userPassword
userPassword: <currentPassword>
-
add: userPassword
userPassword: <newPassword>
-
```

The following example uses the `ldappasswordmodify` tool with the `--passwordChangeMethod ldap-modify` argument to change a user's password:

```
$ bin/ldappasswordmodify --hostname server.example.com --port 636 --useSSL \
  --bindDN "cn=admin,dc=example,dc=com" --bindPassword <bindPassword> \
  --userIdentity uid=jdoe,ou=People,dc=example,dc=com \
  --oldPassword <currentPassword> \
  --newPassword <newPassword> \
  --passwordChangeMethod ldap-modify
```

> **Note**
>
> In the previous example, the `ldappasswordmodify` tool creates the required modifications for an LDAP modify operation.

Password modify extended operation

Alternatively, you can use the password modify extended operation, as described in RFC 3062, to update user passwords. There are two implementation methods:

- For integration with a custom user account management application, use the UnboundID LDAP SDK for Java or another LDAP client API.
- For the server CLI implementation, use the `ldappasswordmodify` tool with the `--passwordChangeMethod password-modify-extended-operation` argument.

Advantages over LDAP modify

The password modify extended operation has several advantages over a regular LDAP modify operation, including:

- The user doesn't have to know their full DN or the name of the attribute used to store their encoded password.
- If the user's password policy is configured with `allow-expired-password-changes` set to `true`, the user can reset their expired password.

  > **Important**
  >
  > This type of request must include a value for `userIdentity`.

- The server can automatically generate a new password for the target user.

- A user can recover access to their account by providing a server-generated password reset token instead of their current password.

> **Note**
> You need to configure the server to support this operation.

Using the extended operation

The following table describes the attributes related to the password modify extended operation:

Attribute	Description
`userIdentity`	Indicates the user whose password you are changing. You can supply one of the following values: • The full DN of the user entry, with or without the `dn:` prefix • The string `u:<user_value>`, where `<user_value>` represents a value that the server can use to return a single user entry The server uses the identity mapper specified in the password modify extended operation handler to match the `u` value to the user entry. The server rejects the password modify request if a search returns more than one entry. By default, the extended operation handler's identity mapper expects an exact match of either the `uid` or `mail` attributes. For example, if you provide the argument `--userIdentity u:jdoe`, the server searches using a filter of `"(|(mail=jdoe)(uid=jdoe))"`. **Tip** If the connection making the password modify request is authenticated as the target user, you can omit `userIdentity`.
`oldPassword`	Indicates the current user password.
`newPassword`	Indicates the new user password. **Tip** To have the server generate a new password, omit `newPassword` from the request. The server uses the password generator defined in the password policy and returns the new password in the extended operation response.

The following example uses `ldappasswordmodify` to target a user by DN and set a new password:

```
$ bin/ldappasswordmodify --hostname ds.example.com --port 636 --useSSL \
  --bindDN "uid=admin,dc=example,dc=com" --bindPasswordFile admin-password.txt \
  --userIdentity uid=jdoe,ou=People,dc=example,dc=com \
  --oldPassword <currentPassword> \
  --newPasword <newPassword> \
  --passwordChangeMethod password-modify-extended-operation
```

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 PingDirectory 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 PingDirectory 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.

> **Tip**
>
> You can find details about configuring the multi-part email account status notification handler in the `config/sample-dsconfig-batch-files/enable-email-account-status-notifications.dsconfig` batch file and about the options for customizing the email templates in the `config/account-status-notification-email-templates/README.txt` file.

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

The PingDirectory server also provides support for several 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 the 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.

The 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 the 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 PingDirectory 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 several 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.

The verify password extended operation

There is an extended operation to verify a user's password. Read carefully before enabling this operation.

For security reasons, PingDirectory provides vague responses to bind requests. If an authentication attempt is unsuccessful, the server doesn't communicate the reason for the failure. Additionally, for both security and performance reasons, the server doesn't attempt to verify the provided credentials if it can determine that the target account is not in a usable state (for example, if it has been administratively disabled, or if it has been locked as a result of too many previous authentication failures).

The `verify-password` extended operation enables you to differentiate the authentication response to the client with respect to certain kinds of failures. For example, if you can determine that a user's account has been locked as a result of too many failed authentication attempts, you can present the user with a means of resetting the password if and only if you can verify that the provided password was correct.

> **Warning**
>
> Enabling the `verify-password` extended operation circumvents server security functionality.
> An attacker could use this extended operation to get unlimited attempts to guess a user's password, circumventing the security benefit of an account lockout. We strongly discourage designing applications in a way that could provide a malicious client with information that might help them better craft an attack against user accounts.
> This extended operation is not a standalone alternative to authenticating clients. Only use it to augment the existing authentication process, if at all.

The `verify-password` extended operation determines if a provided password is correct for a specified user without performing any other password policy processing for that user. This verification is available even if the target account isn't currently allowed to authenticate, and it won't cause any updates to the user's entry as a result of the determination.

Restrictions and privileges

Because of the security risks associated with using `verify-password`, there are a number of restrictions in place to ensure that it can only be used by authorized clients, including:

- **The extended operation handler isn't defined or enabled by default.**
- **The server's access control configuration doesn't permit the use of the extended operation by default. If a client without the `bypass-acl` privilege has a legitimate need to use this extended operation, you must to create a global ACI that grants the client permission to use the extended request with OID 1.3.6.1.4.1.30221.2.6.72.**
- **Requesters must also have the `permit-verify-password-request` privilege, which isn't granted to users (even root users) by default.**
- **Clients must issue `verify-password` requests over a secure connection so that anyone able to view network messages can't access the content of that communication.**

  > **Note**
  >
  > You can disable this safeguard if necessary, but the server still requires secure communication protocol if either the target user's password policy or operational attributes in the user's entry require secure authentication.

- **The extended operation supports clients submitting password verification requests directly to PingDirectory or forwarded through a PingDirectoryProxy server (in both simple-proxy and entry-balanced configurations). When requests are forwarded from PingDirectoryProxy to PingDirectory, you must configure both server instances to enable the extended operation.**

Enabling the verify password extended operation

Before you begin

You must enable the required access privileges described in Restrictions and privileges.

About this task

To verify a password using the `verify-password` extended operation, enable the extended operation and then send a client request with the required information.

The client request must be a JavaScript Object Notation (JSON) object containing the following required fields:

dn

The distinguished name (DN) of the user account whose password the server should verify.

password

The password to verify for that user.

The server response contains a result code that explains the outcome of the `verify-password` operation. The following table describes the result codes that the server can return:

Server response code	Description
`PROTOCOL_ERROR (2)`	The extended request is malformed.
`COMPARE_FALSE (5)`	The provided password isn't correct for the target user.
`COMPARE_TRUE (6)`	The provided password is correct for the target user.
`CONFIDENTIALITY_REQUIRED (13)`	The server is required to only permit the extended operation over a secure connection, or the server is configured to require secure authentication for the target user, but the client is using an insecure connection.
`NO_SUCH_OBJECT (32)`	The target user account does not exist.
`INVALID_DN_SYNTAX (34)`	The provided DN is malformed.
`INAPPROPRIATE_AUTHENTICATION (48)`	The target user account doesn't have a password.
`INSUFFICIENT_ACCESS_RIGHTS (50)`	The client doesn't have sufficient permission to use the extended operation.
`OTHER (80)`	The server encountered an internal error while attempting to verify the password.

Steps

1. Enable the `verify-password` extended operation.

   Example:

   ```
   $ bin/dsconfig create-extended-operation-handler \
     --handler-name "Verify Password" \
     --type verify-password \
     --set enabled:true
   ```

2. Send a properly formed client request.

> **Note**
>
> Learn more about formulating a request using the extended operation.

3. Use the result code in the response to determine how the client responds to the user.

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 the 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:
    - `extop` — Provides a list of extended request OIDs to which the ACI applies.
    - `requestcriteria` — Determines whether an access control rule applies to an operation based on whether that operation matches a given request criteria definition.
    - `target` — Provides an LDAP URL whose distinguished name (DN) identifies the base of the subtree to which the ACI applies.
    - `targetattr` — Provides a list of attributes to which the ACI applies.
    - `targetcontrol` — Provides a list of request control OIDs 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.
    - `targetscope` — Specifies the scope, relative to the target base DN, to which the ACI applies.
2. An opening parenthesis.
3. The string `version 3.0` to indicate the ACI syntax version. The 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:
    - `authmethod` — Identifies the requester by the type of authentication that they used.
    - `connectioncriteria` — Allows or denies an operation based on whether the client connection matches a given connection criteria definition.
    - `dayofweek` — Identifies the requester by the current day of the week.
    - `dns` — Identifies the requester by the resolved name of the client system.
    - `groupdn` — Identifies the requester by group membership.
    - `ip` — Identifies the requester by the IP address of the client system.

- `oauthscope` — Identifies the requester by the set of OAuth scopes that they have.
- `secure` — Allows or denies an operation based on whether the client is communicating with the server over a secure connection; for example, using LDAPS or StartTLS over LDAP.
- `timeofday` — Identifies the requester by the current time of day.
- `userattr` — Identifies the requester by their relation to the value of a specified attribute.
- `userdn` — Identifies the requester by DN or by a predefined keyword that is interpreted by the server.

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.

`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 can 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 would be 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.

requestcriteria

The `requestcriteria` ACI target determines whether an access control rule applies to an operation based on whether that operation matches a given request criteria.

If present in an access control rule, the operator must be either "=" or "!=". The value must be enclosed in quotation marks and it must be the name or full DN of the configuration object that defines the desired request criteria.

For example, let's say that you want to allow members of the `cn=Sales Administrators,ou=Groups,dc=example,dc=com` group to be able to read from and write to the entries for the users that are members of the `cn=Sales Employees,ou=Groups,dc=example,dc=com` group. To do this, you must first create a request criteria object that will match entries for users in the `cn=Sales Employees,ou=Groups,dc=example,dc=com` group. You can do this with the following configuration change:

```
dsconfig create-request-criteria \
     --criteria-name "Requests Targeting Sales Employees" \
     --type simple \
     --set "any-included-target-entry-group-dn:cn=Sales Employees,ou=Groups,dc=example,dc=com"
```

With that request criteria defined, you can use a modification like the following to create the corresponding access control rule:

```
dn: dc=example,dc=com
changetype: modify
add: aci
aci: (targetattr="*")(requestcriteria="Requests Targeting Sales
  Employees")(version 3.0; acl "Allow sales administrators to manage
  sales employees"; allow (read,search,compare,write)
  groupdn="ldap:///cn=Sales Administrators,ou=Groups,dc=example,dc=com";)
```

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` permission for at least one attribute 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.

Changing the allow add ACI behavior for entries

You can require a bind user to have `allow add` permissions for all of an entry's attributes before allowing them to add the entry to PingDirectory.

About this task

By default, a bind user can add an entry to PingDirectory if they have `allow add` permissions for at least one of the attributes in the entry. To increase your control over who is allowed to add entries to your PingDirectory datastore, you can enable the `evaluate-target-attribute-rights-for-add-operations` property.

Enabling this property causes PingDirectory to require a bind user to have an `allow add` access control instruction (ACI) for each attribute of the entry in the add request. If the bind user doesn't meet this condition, or has a `deny add` ACI for any target attributes of the entry to be added, PingDirectory denies the add operation.

> **Warning**
>
> The `evaluate-target-attribute-rights-for-add-operations` property is disabled by default. Enabling this property causes PingDirectory to evaluate the `targetattr` portion of an access control rule for add operations. Before enabling this property in a production environment, you should thoroughly test your existing access control configuration. You might discover cases where you need to add or augment access control rules to ensure that your authorized bind users can continue to add entries as expected.

Steps

- Modify the `evaluate-target-attribute-rights-for-add-operations` property.

  Choose from:

  - Enable the property.

    ```
    $ bin/dsconfig set-access-control-handler-prop \
      --set evaluate-target-attribute-rights-for-add-operations:true
    ```

  - Disable the property.

    ```
    $ bin/dsconfig set-access-control-handler-prop \
      --set evaluate-target-attribute-rights-for-add-operations:false
    ```

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 Lightweight Directory Access Protocol (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.

`connectioncriteria`

The `connectioncriteria` bind rule allows or denies an operation based on whether the client connection matches a given connection criteria definition.

If present in an access control rule, the operator must be either "`=`" or "`!=`". The value must be enclosed in quotation marks, and it must be the name or full DN of the configuration object that defines the desired connection criteria.

For example, you can use a modification like the following to allow any client matching the "Root Users and Topology Administrators" connection criteria to have full read and write access to entries below `dc=example,dc=com`:

```
dn: dc=example,dc=com
changetype: modify
add: aci
aci: (targetattr="*")(version 3.0; acl "Full read and write access for
  administrators"; allow (read,search,compare,write)
  connectioncriteria="Root Users and Topology Administrators";)
```

secure

The `secure` bind rule allows or denies an operation based on whether the client is communicating with the server over a secure connection; for example, using LDAPS or StartTLS over LDAP.

If present in an access control rule, the operator must be either "`=`" or "`!=`", and the value must be either `"true"` or `"false"` including the quotation marks. With this in mind, the `secure` bind rule takes the following forms:

- `secure="true"` indicates that the ACI will apply only if the connection was received over a secure connection.
- `secure="false"` indicates that the ACI will only apply if the connection was received over a non-secure connection.
- `secure!="true"` indicates that the ACI will apply only if the connection was received over a non-secure connection.
- `secure!="false"` indicates that the ACI will apply only if the connection was received over a secure connection.

For example, you can use a modification like the following to allow users below `dc=example,dc=com` to update their own passwords over a secure connection:

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

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.

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 might 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. The 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

The PingDirectory server defines several 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.
`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.

Privilege	Description
`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.
`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.
`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.

Property	Description
`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.
`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.

Property	Description
`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.
`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.

Property	Description
`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.**

Property	Description
`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.
`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

The 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 can 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 1000 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.
`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 5000 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.

Property	Description
`maximum-concurrent-connections`	The maximum number of connections that can be established to the server at any one time. After this limit has been 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. After 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. After 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.
`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 200 messages of the same type in a five-minute period.

Property	Description
`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.

Attribute	Description
`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.

Property	Description
`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. After 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. After 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.
`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

The 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. The 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 the PingDirectory server itself, but are interacting with applications that use the server behind the scenes, then the application could obtain an OAuth token for that user, and then authenticate to the 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 the 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, the 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

The 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 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. The contents of the `config/sample-dsconfig-batch-files/create-topology-admin-user.dsconfig` batch file are shown below:

```
# Although setup automatically creates an initial root user account that can
# be used to manage the server, we strongly recommend creating a separate
# account for each administrator rather than using a single shared account.
# This offers several benefits, including:
#
# * It is easier to determine which administrator performed a given action.
# * There is no need to share credentials or coordinate password changes.
# * It is easier to use strong authentication options like certificates or
#   two-factor mechanisms.
# * You can customize the level of access that each administrator has.
#
# We also recommend creating topology admin user accounts rather than root
# user accounts.  If you create a root user account, then it is only created
# in that one instance.  If you create a topology admin user, then that
# account will be available in all instances in the topology.
#
# Because administrative accounts are very powerful, they are high-priority
# targets for attackers to try to compromise.  They should be required to have
# strong credentials, and you may wish to require that they use strong
# authentication mechanisms (see other dsconfig batch files for configuring
# password validators and other password policy features).  You may also want
# to give them usernames that are not likely to be guessed by an attacker,
# even if they know information about the individuals administering the
# service.  For example, you may not want to use their name in the DN or
# username, choosing instead something that is less predictable or even
# completely random like a UUID.
#
# Once each administrator has their own account, we recommend that you disable
# or remove the initial root user created during setup.  See the
# disable-or-remove-the-initial-root-user.dsconfig batch file for more
# information about that.

#
# NOTE:  Do not edit this file directly.  Changing this file could prevent it
# from being updated during a server upgrade.  If you want to alter the
# dsconfig commands below before applying the configuration changes, copy this
# file to another directory and edit that copy.
#

# Create a new topology administrator account.  In this example, the user will
# be able to read the configuration but will not be allowed to update it, and
# will be given the bypass-read-acl privilege rather than bypass-acl, which
# only ensures they will be able to read entries but they will not be
# permitted to update them unless permitted by the server's access control
# configuration.  They will also only be permitted to communicate with the
# server over a secure connection, and they will only be permitted to
# authenticate with the EXTERNAL or UNBOUNDID-TOTP mechanisms.
dsconfig create-topology-admin-user \
     --user-name "[USER_NAME]" \
     --set "password:[PASSWORD]" \
     --set "first-name:[FIRST_NAME]" \
     --set "last-name:[LAST_NAME]" \
     --set "user-id:[USER_NAME]" \
     --set inherit-default-root-privileges:true \
```

```
    --set privilege:bypass-read-acl \
    --set privilege:-bypass-acl \
    --set privilege:-config-write \
    --set search-result-entry-limit:0 \
    --set time-limit-seconds:0 \
    --set look-through-entry-limit:0 \
    --set idle-time-limit-seconds:0 \
    --set require-secure-authentication:true \
    --set require-secure-connections:true \
    --set "allowed-authentication-type:SASL EXTERNAL" \
    --set "allowed-authentication-type:SASL UNBOUNDID-TOTP"
```

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

The 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.

The 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.

The 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 The 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.

The 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 PingDataSync 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. The 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 the 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 the 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.

The PingDirectory server offers the following types of debug loggers:

- A file-based debug log publisher that records messages 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 the 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

The 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.

The 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 after 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. The 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.

The 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

The 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

The 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 the 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

The 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

The 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 several 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

The 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 several 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 several 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.
`changelog-deleted-entry-exclude-attribute`	Specifies which attributes should be excluded from changelog entries for delete operations.

Property	Description
`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

The 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

The 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 PingDirectoryProxyProxy server
- Information about load balancing state in PingDirectoryProxy server
- Information about entry balancing state in PingDirectoryProxy server
- Information about the 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 such as Kubernetes, you can 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. In this case,you can call the availability servlet in a liveness or readiness probe.

To assist with this, the 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

The 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 the 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 the PingDirectoryProxy server changes the assessed health check state for any of its backend servers
- If the PingDirectoryProxy server encounters a problem while performing entry rebalancing processing
- If the 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 the 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

The 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 the 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.

> **Important**
>
> SNMP is deprecated.

StatsD endpoints

The 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.

The 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. The 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 the 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

The 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

The 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

The 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

The 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 the PingDirectory server do the following:

- 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.
- Identify all accounts with password policy state issues that can currently or soon affect their usability.
- Identify all accounts with activation times in the future, expiration times in the past, or expiration times in the near future.
- Identify accounts with passwords encoded using a deprecated password storage scheme.
- Identify accounts for users that have not authenticated in longer than a specified length of time.
- Identify accounts that are configured to use a nonexistent password policy (and are therefore unable to authenticate).
- Identify accounts that match a specified search filter.

> **Note**
>
> You can also use the PingDirectory server SDK to develop custom data security auditors for identifying other kinds of issues.

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, the PingDirectory server provides support for soft deletes, in which the server preserves the entry instead of removing it. 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.

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.

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`.

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.

Consent Solution Guide

This guide provides information for configuring and managing the PingDirectory server's Consent Service, which supports end-user control of personal data by collecting authorization from users and customers.

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.

For example, 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, the 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 by the PingDirectoryProxy server.

Enterprises can integrate the following Consent Service features into their applications to give users transparency and control of their data privacy:

- The collection of consent from application users
- The enforcement of consent
- A user's management of his or her consent
- Auditing of consent actions

The following terms are used throughout this guide:

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, or 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

PingDirectory and PingDirectoryProxy provide a REST application programming interface (API) for managing an individual's consent to handle their data.

The PingDirectory Consent API enables the following authorization services for managing an individual's consent and data:

- Captures a user's consent for sharing or processing data
- Confirms a user's consent to share or process data has been granted
- The user can manage the consent that they have granted

The Consent REST API can be used as a component of a larger solution, such as a General Data Protection Regulation (GDPR) compliance system.

For more information on the Consent API, see the PingDirectory Consent API Reference and How applications use the Consent API.

How consents are collected

The Consent API allows the user to make the decision about sharing their data with other applications through a consent record.

User consent is collected by creating a consent record through the Consent API. In most cases, the Consent API client uses the consent localization data to construct an approval prompt that displays 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 uses the Consent API to look up the localizations for the `browsing-behavior` consent definition. It selects 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 stores the decision by creating a new consent record through the Consent API.

How consents are enforced

The Consent API can be used as a data source for making access control decisions and enforcing the user's consent.

If a data usage scenario requires consent, then the application or service can't process or access the data unless the user has provided consent. The entity that performs this consent check can be the application itself or some other service.

To perform a consent check, the Consent API client tries 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, it can use the `browsing-behavior` consent definition. In this data collection scenario, the application searches the Consent API and checks the an existing consent grant for a consent record that matches the user and the `browsing-behavior` consent definition. If a match is found, then the application proceeds. 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 using the Consent API.

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.
2. The application wants to record the page visit but first checks if the user has granted consent to do so.
3. 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.
4. The API returns a result indicating that no consent record exists.
5. The application prompts the user for his or her consent.
6. 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.
7. After the user makes a decision, the application stores the user's decision by creating a new consent record through a call to the Consent API.
8. Later, the user visits another page in the company's site, and the application wants to record the page visit, so it checks whether the user has granted consent to do so.
9. The application makes a call to the Consent API to get the `browsing-behavior` consent record for this user.
10. 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.

Configuring the Consent Service

This section provides information for installing and configuring the components to support the Consent Service.

Installing and configuring the Consent Service includes the following topics:

- Configuration overview
- Example configuration scenarios
- Setting up with the configuration scripts
- Setting up in a replicated PingDirectory server environment
- Configuration reference

- Authorization

For more configuration information, see PingDataSync Server Administration Guide.

Configuration overview

Your Consent Service configuration can vary depending on client application capabilities, authentication methods used, and other factors.

By default, the Consent Service is not enabled. The setup and configuration process varies depending on the following factors:

- Whether client applications allow an individual to self-manage consents
- Whether some or all client applications can 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

Review the following client application scenarios to determine how to configure the Consent Service 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's a client application for consent administrators.

An OAuth 2 authorization server grants access tokens that the applications use 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 Lightweight Directory Access Protocol (LDAP) distinguished names (DNs) used by the PingDirectory server.
3. Configure 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 can't.

The following configuration steps describe the setup needed for the PingDirectory 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 the consent record subject and actor attribute values to LDAP DNs.
4. Configure the Consent Service to use the application's service account.
5. Configure the Consent Service to use the identity mapper.

Setting up with the configuration scripts

The PingDirectory server includes two configuration scripts that can serve as the starting point for setting up the Consent Service.

About this task

You can use one of the following two configuration scripts to start setting up the Consent Service:

- `consent-service-base-entries.ldif` - Import this LDIF script to create the base distinguished name (DN) where consent records are stored.
- `consent-service-cfg.dsconfig` - Import this script to configure and enable the Consent Service.

Both scripts are located in the `/resource/consent/` directory of the PingDirectory server root.

> **Tip**
>
> Carefully review and update both scripts to support your client application scenarios and business needs.

Steps

1. Choose a configuration script to set up the Consent Service.

   Choose from:

   - Basic configuration with the `consent-service-base-entries.ldif` file.
     1. Edit the LDIF script and change the location of where you want to store the consent records.
     2. Import the LDIF script using the `ldapmodify` command. See the following example.

        ```
        $ bin/ldapmodify --defaultAdd \
          --filename consent-service-base-entries.ldif
        ```

   - Basic configuration with the `consent-service-cfg.dsconfig` file.
     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. See the following example.

```
$ bin/dsconfig --no-prompt \
  --batch-file consent-service-cfg.dsconfig
```

Setting up in a replicated PingDirectory server environment

Running the Consent Service setup script differs depending on if your environment includes replicated PingDirectory servers.

About this task

Setting up the Consent Service requires different steps depending on if replication is enabled in your environment or not.

> **Tip**
>
> If possible, you should set up the Consent Service after replication is enabled for the PingDirectory servers.

You can find more information about server replication in the PingDirectory Server Administration Guide.

Steps

1. Choose a Consent Service setup:

   Choose from:

   - If you are setting up the Consent Service after replication is enabled for PingDirectory servers:
     1. Configure the PingDirectory servers to use a configuration group called `all-servers`.

        > **Note**
        >
        > This ensures 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
        ```

   - If you have set up the Consent Service on a standalone PingDirectory server before enabling replication:

> **Note**
>
> In this example, DS1 is the original PingDirectory server, and DS2 is the second server to 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

Use this section as a reference for information about the Consent Service properties and configuration.

There are many configuration options for the Consent Service and application integration. The configuration scripts included with PingDirectory server provide a starting point. Additional 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.

You configure the Consent Service properties using the `dsconfig set-consent-service-prop` command. You can use the Consent Service configuration script to configure the Consent Service properties, as show in the following example.

```
$ 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"
```

Consent Service properties

Property	Description	Required to enable service
`enabled`	If set to `true`, the property enables the Consent Service for handling client requests.	Yes
`base-dn`	Specifies a container distinguished name (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 are considered privileged when using basic authentication. If not defined, a requester is only considered privileged if it's mapped to a DN with the `bypass-acl` privilege.	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.	No
`audience`	Specifies an `audience` claim value that the Consent Service requires to be present in bearer tokens that it accepts.	No

For the Consent Service to report itself as available to clients:

- 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 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

Each consent record is a distinct entry in the PingDirectory server. The Consent Service requires that these entries are stored under a common base distinguished name (DN).

About this task

The `base-dn` property of the Consent Service configuration defines the base DN. The Consent Service LDIF file sets the base DN.

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 `consent-service-base-dn.ldif` file.

   Example:

   ```
   dn: ou=consents,dc=example,dc=com
   objectClass: top
   objectClass: organizationalUnit
   ou: consents
   ```

2. Use `ldapmodify` to add the entry.

   Example:

   ```
   $ bin/ldapmodify --defaultAdd --filename consent-service-base-dn.ldif
   ```

Creating an internal service account

Create an internal LDAP connection to operate against consent records that are stored as LDAP entries.

About this task

The Consent Service uses an internal LDAP connection to operate against consent records that are stored as LDAP entries. The Consent Service authenticates the LDAP connection using a service account that must be created and dedicated solely to the Consent Service.

The Consent Service configuration script configures the internal service account using a topology administrator user. If needed, this can be changed to a root distinguished name (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.
- 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 Managing access control.

Steps

1. To ensure the correct access, create a user with the `bypass-acl` privilege.

   Example:

   The following `dsconfig` command creates a topology admin user with the `bypass-acl` privilege.

   ```
   $ 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
   ```

   Tip

   The `bypass-acl` privilege grants a broad level of access, so you might not want to grant this privilege to the Consent Service account.

2. Set this user as the `bind-dn` for the Consent Service.

3. To enable a targeted set of functionality for the Consent Service, add the following access control instruction (ACI).

   Example:

   The following example grants the access to the `cn=consent service account` DN 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";)'
   ```

Configuring an identity mapper

The Consent Service uses identity mappers to map requester identities, subject values, and actor values to distinguished names (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 mapper types and descriptions

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 sets the `subject`, `subjectDN`, `actor`, and `actorDN` values to the identity of the authenticated requester. If the requester uses basic authentication, then all values are 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 are set to the auth DN determined by the access token validator identity mapper.

Privileged clients can manually set a consent record's `subject` and `actor` values. In those cases, the Consent Service's `consent-record-identity-mapper` property is used to map a consent record's `subject` and `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 matches a user identifier to an Lightweight Directory Access Protocol (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
```

This configuration shows another typical example: an identity mapper that matches 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
```

This final example creates an identity mapper that matches a user identifier to an LDAP entry with the same value in either its `uid`, `cn`, or `entryUUID` attribute. This identity mapper also constrains its search to the `cn=config` and `ou=people,dc=example,dc=com` and **`cn=config`** base DNs. By default, the `cn=config` base DN is not searched 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 the following HTTP authentication methods and the use of token validators for authorization.

Supported HTTP authentication methods

The Consent Service supports the following HTTP authentication methods that are enabled by default:

Basic authentication

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 distinguished name (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 Lightweight Directory Access Protocol (LDAP) bind using the DN and password to determine if the request can be processed.

Bearer token authentication

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 Consent servlet looks at the request's Authorization header to determine which authentication type is being used by the client.

Configuring token validators

The PingDirectory server must be configured to accept access tokens using one or both of the following 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 check boxes mentioned in steps 2 and 3 in Enabling the OAuth AS role in the PingFederate documentation.
> Starting with PingFederate 10.1, these items are always enabled.

JWT access token validator

Supports signed or encrypted JSON Web Token (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, the Consent Service uses this access token validator. 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 the following 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.
>
> See the authorization server's documentation for guidance.

Configuring basic authentication

Disable or enable basic authentication and configure an identity mapper for basic authentication.

About this task

By default, basic authentication is enabled. The settings are configured in the Consent HTTP Servlet Extension configuration.

> **Note**
>
> All of these configuration changes require the Consent servlet to be reloaded before they can take effect. To do this, see the last step.

Steps

- **To disable basic authentication, use the following command.**

  Example:

  ```
  $ bin/dsconfig set-http-servlet-extension-prop \
    --extension-name Consent \
    --set basic-auth-enabled:false
  ```

- **To enable basic authentication, use the following command.**

  Example:

  ```
  $ bin/dsconfig set-http-servlet-extension-prop \
    --extension-name Consent \
    --set basic-auth-enabled:true
  ```

- **To configure an identity mapper for basic authentication, use the following command.**

  Example:

```
$ bin/dsconfig set-http-servlet-extension-prop \
  --extension-name Consent \
  --set "identity-mapper:User ID Exact Match"
```

- To restart the connection handler that hosts the Consent servlet, use the following commands.

  Example:

```
$ 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

Configure an access token validator.

About this task

> **Note**
>
> You can configure the Consent Service to use a single validator.

Steps

- Configure an access token validator using `dsconfig`.

  Example:

  This example shows an access token validator configured on a 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
```

- (Optional) If more than one access token validator is configured on a PingDirectory server, you can configure the Consent Service to use a single validator with the following command.

  Example:

  ```
  $ bin/dsconfig set-http-servlet-extension-prop \
    --extension-name Consent \
    --set "access-token-validator:PingFederate Token Validator"
  ```

Configuring Consent Service scopes

Configure the `privileged-consent-scope` and `unprivileged-consent-scope` for the Consent Service.

About this task

The Consent Service checks access tokens for a `subject` claim and uses an identity mapper to map the value to a distinguished name (DN), called the request DN or auth DN. If no request DN can be mapped, the request is rejected.

The Consent Service only accepts an access token with a scope that it is configured to recognize.

`unprivileged-consent-scope`

An unprivileged consent scope designates the requester as unprivileged. The scope's name is configured with the Consent Service's `unprivileged-consent-scope` property.

`privileged-consent-scope`

A privileged consent scope designates the requester as privileged. This is configured using the Consent Service's `privileged-consent-scope` property.

> **Note**
>
> The authorization server must also be configured to issue tokens with these scopes.

Steps

- Configure the `privileged-consent-scope` and `unprivileged-consent-scope` for the Consent Service.

  Example:

  ```
  $ 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 distinguished name (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
`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 the bearer token doesn't contain a scope matching the Consent Service's `privileged-scope-name` property, 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 Lightweight Directory Access Protocol (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 the `auth DN` isn't listed in the Consent Service's `service-account-dn` property, the requester is considered `unprivileged`.

Managing Consents

This section provides information about the collection and end-user control of personal data as well as managing users' consents.

Managing users' consents includes the following topics:

- Overview of consent management
- Consent definitions and localizations
- Perform an audit on consents
- Logging
- Correlating user and consent data
- Troubleshooting the Consent Service

Overview of consent management

Consent management involves more than collecting the user's consent.

In the consent management life cycle:

1. The terms of each consent contract are centrally managed.
2. The user reviews previously granted consents and potentially revokes some of them.
3. Companies trace the history of updates to any consent in order to resolve a dispute or respond to audit.

Consent definitions and localizations

The Consent Service requires consent definitions and localizations, which collect and share a use's data and define the purpose for collecting and sharing this data.

Companies can centrally manage the language used when prompting a user to give consent to ensure 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 the data

For example, a consent definition could represent users' email addresses that are used to deliver a third party's email newsletter. A consent definition could also represent access to a user's network-connected IoT device that could be used for a home automation task controlled by a third party.

Each consent definition must have one or more localizations. 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 a consent definition and localization

Create and update a consent definition and localization.

Steps

- Create a consent definition using `dsconfig create-consent-definition`.

  Example:

  ```
  $ bin/dsconfig create-consent-definition \
    --definition-name email_newsletter \
    --set "display-name:Email newsletter"
  ```

- Create a localization for the consent definition using `dsconfig create-consent-definition-localization`.

  Example:

  ```
  $ 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"
  ```

- Update a localization and the version using `dsconfig set-consent-definition-localization-prop` and `set version`.

  Example:

  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

The Consent Service offers two types of audit logs to track changes and to perform audits on Consent Service resources.

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, which 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`.
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 `CONSENT AUDIT` message type, and a set of key/value pairs.

> **Note**
>
> The keys used in trace log audit messages vary depending on the type of resource.

The following table describes a subset of important keys.

Trace logger key	Description
`requestID`	A server-specific HTTP request ID. This value can be correlated with messages produced by other loggers.

Trace logger key	Description
`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 distinguished name (DN) of the requester, which is available only when the resource type is `consent`.
`definitionID`	The consent definition ID. The following list identifies the possible resource types and their definitions: **`definition`** Identifies the definition that was changed. **`localization`** Identifies the parent definition. **`consent`** Identifies the consent record's related definition.
`locale`	The locale. The following list identifies the possible resource types and their definitions: **`localization`** Identifies the localization in combination with the definition ID. **`consent`** 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. This is available only when the resource type is `consent`.
`actor`	The actor value. This is available only when the resource type is `consent`.
`actorDN`	The actor's mapped LDAP DN. This is available only when the resource type is `consent`.

Trace logger key	Description
`audience`	The audience value. This is available only when the resource type is `consent`.
`status`	The consent status. This is only available when the resource type is `consent`. Possible values are `pending`, `accepted`, `denied`, `revoked`, and `restricted`.
`previousStatus`	The previous consent status, if applicable. This is 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 is 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 distinguished name (DN) is known, the `subjectDN` key can be used to construct a text search for any audit log messages containing that DN. Any matching log messages constitute a history of that individual's consent activity.

New consent record example

This example shows an audit log message that provides important values in a parseable key/value format and includes a complete 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=example,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':'2018-05-22T23:02:42.553Z'}"
```

Updated consent record example

This example provides a complete consent record before and after it was updated. By reviewing the `attrsUpdated`, `status`, and `previousStatus` keys, you 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=example,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':'2018-05-22T23:02:42.553Z'}
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=People,dc=example,
  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':'2018-05-22T23:05:08.655Z'}"
```

Deleted consent record example

This example shows that a consent record has been deleted and provides a complete representation of the consent record before it was deleted.

```
[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,subjectDN"
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=People,
   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

Use the PingDirectory server trace log publisher for logging events generated by HTTP service operations.

About this task

You can use the trace logger to observe, debug, and audit consent requests.

Steps

- Create a trace logger for all consent events using `dsconfig create-log-publisher`.

> **Tip**
>
> To create a log of consent audit events only, remove all message types except for **`consent-message-type:audit`**.

Example:

The following example creates a trace logger for all consent events and 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

An organization that has been granted consent by a group of users can perform an LDAP search so that they can use the consent data in the aggregate.

About this task

For this task, consider the example scenario where a marketing group has collected consent to send a newsletter by email. To find all the users that have granted consent to receive emails, the marketing group performs a search that lists all of the consent records where the consent definition is `email` and the status is `accepted`. Then, the marketing group must correlate these consent records to user entries and retrieve each user's email address.

Every consent record contains a `subject` field, the user whose data is collected and stored. You can configure the Consent Service so that it stores the subject's distinguished name (DN) in the `subjectDN` field.

Steps

- Perform a search using the `ldapsearch` command.

  Example:

  > **Note**
  >
  > The example includes the following modifications in the `ldapsearch` command:
  > - To correlate the consent record entries to user entries and retrieve each user entry's `mail` attribute value, `ping-consent-subject-dn` is used.
  > - To find all of the relevant consent record entries, the LDAP search specifies values where `ping-consent-definition.id` is `email` and the `ping-consent-status` is `accepted`.

  ```
  $ 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
  ```

  - A consent record's `subjectDN` field is the `ping-consent-subject-dn` attribute.
  - A consent record's status is in the `ping-consent-state` JSON attribute field.
  - A consent record's definition ID is in the `ping-consent-definition.id` JSON attribute field.
  - A user entry's email address is in the `mail` attribute.

    Result:

    The example LDAP search returns the following results.

```
# 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
```

> **Note**
>
> The `Join Result Control:` output specifies the `mail` value.

Troubleshooting the Consent Service

This section provides general guidelines for troubleshooting the Consent Service and any connection issues.

When evaluating the configuration:

- Make sure that the Consent Service is enabled.
- Make sure that the Consent Service base distinguished name (DN) exists.
- Make sure that the Consent Service's service account has the correct permissions.
- If the Consent Service should accept bearer tokens, make sure that:
  - One or more access token validators are configured correctly.
  - The identity mappers for the access token validators are configured correctly.
  - The authorization servers are configured correctly to issue tokens that the Consent Service can accept. Check the `audience`, `privileged-consent-scope`, and `unprivileged-consent-scope` properties of the Consent Service configuration.
- If privileged users are defined, make sure that the members of the Lightweight Directory Access Protocol (LDAP) group are specified by the Consent Service configuration's `privileged-users-group-dn` property.
- If there are applications that allow individuals to manage their own consents, make sure that the system is properly configured to map `actor` and `subject` DNs. Check the Consent Service configuration's `consent-record-identity-mapper` property.

Error cases

The following topic discusses troubleshooting possible error cases and solutions.

Consent Service is unavailable

If the Consent Service is unavailable, check the following:

- Ensure that the service is enabled and that communication 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 servers behind the PingDirectoryProxy server.

Requester lacks sufficient rights to perform operation

A request might 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 can perform. For example, the requester attempted to act upon a consent record that it does not own, or it 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 doesn't 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` is unindexed, perform a search by `subject` and `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 might be able to refine the search to produce fewer results.