UnboundID® Directory Proxy Server Administration Guide

Version 3.6.0

UnboundID Corp
13809 Research Blvd, Suite 500
Austin, Texas, 78750
Tel: +1 512.600.7700
Email: support@unboundid.com

Copyright

Published: 2012-10-29

Contents

Preface

This guide presents the procedures and reference material necessary to install, administer and troubleshoot the UnboundID® Directory Proxy Server in multi-client, high-load production environments.

Purpose of This Guide

The purpose of this guide is to provide valuable procedures and concepts that can be used to manage the UnboundID® Directory Proxy Server in a multi-client environment. It also provides information to monitor and set up the necessary logs needed to troubleshoot the server's performance.

Audience

The guide is intended for administrators responsible for installing, maintaining, and monitoring servers in large-scale, high load production environments. It is assumed that the reader has the following background knowledge:

- Directory Services and LDAPv3 concepts
- System administration principles and practices
- Understanding of Java VM optimization and garbage collection processes
- Application performance monitoring tools

Related Documentation

The following list shows the full documentation set that may help you manage your deployment:

- *UnboundID® Directory Server Administration Guide*
- *UnboundID® Directory Server Reference Guide (HTML)*
- *UnboundID® Directory Proxy Server Administration Guide*
- *UnboundID® Directory Proxy Server Reference Guide (HTML)*
- *UnboundID® Synchronization Server Administration Guide*
- *UnboundID® Synchronization Server Reference Guide (HTML)*
- *UnboundID Metrics Engine Administration Guide*
- *UnboundID LDAP SDK for Java API*
- *UnboundID Server SDK API*

Document Conventions

The following table shows the document convention used in this guide.

Convention	Usage
`Monospace`	Commands, filenames, directories, and file paths
`Monospace Bold`	User interface elements, menu items and buttons
Italic	Identifies file names, doc titles, terms, variable names, and emphasized text

Chapter 1 Introduction

UnboundID® Directory Proxy Server is a fast and scalable LDAPv3 gateway for the UnboundID® Directory Server. The directory proxy server architecture can be configured to control how client requests are routed to backend servers.

This chapter provides an overview of the directory proxy server features and components. It contains the following sections:

Topics:

Overview of the Proxy Server Features

The UnboundID® Directory Proxy Server is a fast, scalable, and easy-to-use LDAP proxy server that provides high availability and additional security for the UnboundID® Directory Server, while remaining largely invisible to client applications. From a client perspective, request processing is the same, whether communicating with the directory server directly or going through the directory proxy server.

The UnboundID® Directory Proxy Server provides the following set of features:

- **High availability**. The directory proxy server allows you to transparently fail over between servers if a problem occurs, as well as ensuring that the workload is balanced across the topology. If a client does not support following referrals, the directory proxy server can follow them on the client's behalf.
- **Data mapping and transformation**. The directory proxy servers can do DN mapping and attribute mapping to allow clients to interact with the server using older names for directory content. The directory proxy server allows clients to continue working when they would not be able to work directly with the directory server, either because of changes that have occurred at the directory layer or to inherent design limitations in the clients.
- **Horizontal scalability and performance**. Reads can be horizontally scaled using load balancing. In large data centers, if the data set is too large to be cached or to provide horizontal scalability for writes, the directory proxy server can automatically split the data across multiple systems. This feature allows the proxy to improve scalability and performance of the directory environment.
- **Load balancing and failover**. You can spread the workload across directory proxy server in a large data center using the directory proxy server's load-balancing algorithms. Load balancing is also useful when a server becomes degraded or non-responsive, because client process requesting is directed to a different server.
- **Security and access control**. The directory proxy server can add additional firewall capabilities, as well as constraints and filtering to help protect the directory server from attacks. You can use a directory proxy server in a DMZ as opposed to allowing clients to directly access the directory server in the internal network or providing the data in the DMZ. It can help provide secure access to the data and you can define what actions clients are allowed to do. For example, you can prevent clients from making modifications to data when connected via a VPN no matter what their identity or permissions.
- **Tracking of operations across the environment**. In the past, administrators have commonly complained that when they see a request in the access log, they have no idea where it came from and cannot track it back to a particular client. The directory proxy server contains controls that allow administrators to track requests back to the client that issued them. Whenever the directory proxy server forwards a request to the directory server, it includes a control in the request so that the directory server's access log has the IP address of the client, address and connection ID of the directory proxy server. In the response back to the client, it similarly includes information about the directory server that processed the request, such as the connection ID and operation ID. This feature makes it easier for administrators to keep track of what is going on in their environment.

- **Monitoring and management tools**. Because the directory proxy server uses many of the components of the UnboundID® Directory Server, it can leverage them to provide protocol support, logging, management tools for configuration and monitoring, schema, and so on. You can use the `dsconfig` tool and the Web-based administration console to manage the directory proxy server.

- **Multi-Platform Support**. The UnboundID® Directory Proxy Server is a pure Java application and is certified VMWare Ready™. It is intended to run within the Java Virtual Machine on any Java Standard Edition (SE) or Enterprise Edition (EE) certified platform. For the list of supported platforms and Java versions, access your Customer Support Center portal or contact your authorized support provider.

  Any known OS or JDK-related issues will be documented in the release notes distributed with the product. Direct any questions or requests for additional platform certifications to your authorized support provider.

Overview of the Directory Proxy Server Components and Terminology

The proxy consists of the following components and functionality that provide the proxy capabilities:

- Locations
- LDAP External Servers
- LDAP Health Checks
- Load-Balancing Algorithms
- Data Transformations
- Request Processors
- Server Affinity Providers
- Subtree Views
- Connection Pools
- Client Connection Policies
- Entry Balancing

This section describes each component in more detail.

About Locations

Locations define a group of servers with similar response time characteristics. Each location consists of a name and an ordered list of preferred failover locations. The directory proxy server and each of the backend LDAP external servers can be assigned locations. These locations can be taken into account when deciding how to route requests, so that the server prefers to forward requests to directory servers in the same data center over those in remote locations. As a rule of thumb, if you have multiple data centers then you should have a separate location for each one. In most environments, all directory proxy server instances should have the same configuration except for the attribute that specifies the location of the directory proxy server itself.

For example, a deployment consists of three data centers, one in New York, another in Chicago, and another in Los Angeles. In the New York data center, applications which reside in this data center prefer communicating with directories in this data center. If none of the servers are available, it prefers to failover to the data center in Chicago rather than the data center in Los Angeles. So the New York location contains an ordered list in which the Chicago location is preferred over the Los Angeles data center for failover.

For information about configuring locations, see *Configuring Locations*.

About LDAP External Servers

You can configure information about the directory server instances accessed by the UnboundID® Directory Proxy Server. This configuration information includes the following:

- Server connection information, such as IP address, port, and security layer
- Location
- Authentication information
- Methods for authenticating and authorizing clients
- Server-specific health checks
- Types of operations allowed. For example, some LDAP external servers may allow only reads and others allow reads and writes, so the proxy server can recognize this and accommodate it.

The UnboundID® Directory Proxy Server allows you to configure different types of LDAP external servers. The default configuration for each type is tuned to be the best possible configuration for each.

For information about configuring LDAP external servers, see *Configuring LDAP External Servers*.

About LDAP Health Checks

The LDAP health check component provides information about the availability of LDAP external servers. The health check result includes a server state, which can be one of the following:

- **Available**. Completely accessible for use.
- **Degraded**. The server may be used if necessary, but has a condition which may make it less desirable than other servers (for example, it is slow to respond or has fallen behind in replication).
- **Unavailable**. Completely unsuitable for use (for example, the server is offline or is missing critical data).

Health check results also include a numeric score, which has a value between 1 and 10, that can help rank servers with the same state. For example, if two servers are available and one has a score of 8 and the other a score of 7, the proxy can be configured to prefer the server with the higher score.

The directory proxy server periodically invokes health checks to monitor each LDAP external server, and may also initiate health checks in response to failed operations. It checks the health of the LDAP external servers at intervals configured in the LDAP server's `health-check-frequency` property. However, the directory proxy server has safeguards in place to ensure that only one health check is in progress at any time against a backend server to avoid affecting its ability to process other requests.

The results of health checks performed by the proxy server are made available to the load-balancing algorithms so that they may be taken into account when determining where to send requests. The directory proxy server will attempt to use servers with a state of available before trying servers with a state of degraded. It will never attempt to use servers with a state of unavailable. Some load-balancing algorithms may also take the health check score into account, such as the health-weighted load-balancing algorithm, which prefers servers with higher scores over those with lower scores. Other load-balancing algorithms do not use the health check scores, such as the round-robin load-balancing algorithm, which balances the load equally among servers with the same state, regardless of the health check score. You configure the algorithms that work best for you environment.

In some cases, an LDAP health check may define different sets of criteria for promoting and demoting the state of a server. So, a degraded server may need to meet more stringent requirements to be reclassified as available than it originally took to be considered degraded. For example, if response time is used in the process of determining the health of a server, then the directory proxy server may have a faster response time threshold for transitioning a server from degraded back to available than the threshold used to consider it degraded in the first place. This threshold difference can help avoid cases in which a server repeatedly transitions between the two states because it is operating near the threshold.

For example, the health check used to measure search response time is configured to mark any server to be marked degraded when the search response time is greater than 1 second. You can then configure that the response time must be less than 500 ms before the server is made available again, so that the directory proxy server does not flip back and forth between available and degraded.

UnboundID® Directory Proxy Server provides the following health checks:

- **Measure the response time for searches and examine the entry contents**. For example, the health check might retrieve a monitoring entry from a server and base the health check result on whether the entry was returned, how long it took to be returned, and whether the value of the returned entry matches what was expected.
- **Monitor the replication backlog**. If a server falls too far behind in replication, then the directory proxy server can stop sending requests to it. A server is classified as degraded or unavailable if the threshold is reached for the number of missing changes, the age of the oldest missing change, or both.
- **Consume directory server administrative alerts**. If the directory server indicates there is a problem, for example it enters lockdown mode, then the directory proxy server will stop sending requests to the server. The directory proxy server detects administrative alerts as soon as they are issued and then checks to see if the alert is associated with a server becoming degraded or unavailable. This health check also confirms if a server is placed in the degraded or unavailable state without an alert being issued.

- **Monitor the busyness of the server**. If a server becomes too busy, then it may be marked degraded or unavailable so that less heavily-loaded servers may be preferred.

For information about configuring health checks, see *Configuring Server Health Checks* on page 61. To associate a health check with an LDAP external server and set the health check frequency, you must configure the `health-check` and `health-check-frequency` properties of the LDAP external server. See “To Configure an External Server Using dsconfig” on page 63 for information about configuring the properties of the external server.

About Load-Balancing Algorithms

Load-balancing algorithms are used to determine which server in a set of similar servers should be used to process a client request. The algorithm can take the following criteria into account:

- Consider the location of the server. Servers in the same location as the directory proxy server can be preferred over those in alternate locations.
- Consider the health of the server. Servers that are available are preferred over those that are degraded. In some cases, the health check score may also be used to further differentiate between servers with the same health check state.
- Route requests consistently. Requests from a single client may be consistently routed to the same directory server instance to avoid problems such as propagation delay from replication.
- Retry the operation in an alternate server if the request fails or the operation times out. You can control if the retry is allowed and, if so, how many times to retry and the time out interval.

The UnboundID® Directory Proxy Server provides the following load-balancing algorithms:

- **Fewest operations**. Requests are forwarded to the backend server with the fewest operations currently in progress.
- **Single server**. Requests are always sent to the same server and will not attempt to fail over to another server if the target server is unavailable.
- **Round robin**. Requests are sent to a given pool of servers.
- **Weighted**. Administrators explicitly assign numeric weights to individual servers or sets of servers to control how likely they are to be selected for processing requests relative to other servers.
- **Health-based weighting**. Uses the health check score to assign weights to each of the servers, so that a server with a higher score gets a higher percentage of the traffic than a server with a lower score. The proportion of traffic received is the difference between their health check scores.
- **Failover**. Requests are always sent to a given server first. If that server fails, then the request is sent to another specified server, and so on through an ordered failover server list.

For information about configuring load balancing, see *Configuring Load Balancing*.

About Proxy Transformations

Proxy transformations are used to rewrite requests and responses as they pass through the directory proxy server. Proxy data transformations are helpful for clients that use an old schema or that contain a hard-coded schema.

Proxy transformations can provide DN and attribute mapping altering both requests to the server as well as responses from the server. For example, a client sends a request to `o=example.com` even though the directory server handling the request uses `dc=example,dc=com`. The directory proxy server can transparently remap the request so that the server can process it, and map it back to the original DN of the client request when the value is returned. Or if a client tries to use the attribute `userID`, the directory proxy server can map it to `uid` before sending the request on to the backend LDAP server. The directory proxy server then remaps the response to `userID` when the value is returned.

The directory proxy server also includes a proxy transformation that can be used to suppress a specified attribute, so that it will never be returned to clients. It can also cause the server to reject requests which target that particular attribute. Another proxy transformation can be used to prevent entries that match a given search filter from being returned to clients.

For information about configuring proxy transformations, see *Configuring Proxy Transformations* on page 70.

About Request Processors

A request processor encapsulates the logic for handling an operation, ensuring that a given operation is handled appropriately. The request processor can either process the operation directly, forward the request to another server, or hand off the request to another request processor.

UnboundID® Directory Proxy Server provides the following types of request processor:

- **Proxying request processors**, which forward operations received by the directory proxy server to other LDAP external servers.
- **Entry-balancing request processors**, which split data across multiple servers. They determine which set of servers are used to process a given operation. They then hand off operations to proxying request processors so that requests can be forwarded to one of the servers in the set.
- **Failover request processors**, which perform ordered failover between other types of request processors, sometimes with different behavior for different types of operations. For example, you could use a failover request processor to achieve round-robin load balancing for read operations but failover load-balancing for writes.

Proxy server request processors can be used to forward certain controls, including the batch transaction control and the LDAP join control. The batch transaction control must target a single Berkley DB backend. For more information about the controls, refer to the UnboundID LDAP SDK for Java documentation.

For information about configuring request processors, see *Configuring Request Processors* on page 72.

About Server Affinity Providers

The server affinity provider can be used to establish an affinity to a particular backend server for certain operations. You can configure one of three types of provider:

- Client connection server affinity, so that requests from the same client connection may consistently be routed to the same backend server.
- Client IP address server affinity, so that all requests coming from the same client system will be consistently routed to the same backend server.
- Bind DN server affinity, so that all requests from the same user will be consistently routed to the same backend server.

For information about configuring server affinity, see *Configuring Server Affinity*.

About Subtree Views

A subtree view can be used to make a portion of the DIT available to a client by associating a request processor with a base DN. Subtree views allow you to route operations concerning one set of data to a particular set of data sources, and operations concerning another set of data to another set of data sources. Multiple subtree views may be involved in processing a request, such as for searches that have a scope that is larger than the subtree view.

The subtree view includes a single base DN used to identify the portion of the DIT. They may have hierarchical relationships, for example one subtree view could be configured for `dc=example,dc=com` and another for `ou=People,dc=example,dc=com`.

For information about configuring a subtree view, see *Configuring Subtree Views*.

About the Connection Pools

Based on the type of backend server that you are using, the UnboundID® Directory Proxy Server maintains either one or two connection pools to the backend server. It maintains either one pool for all types of operations or two separate pools for processing bind and non-bind operations from clients. When the directory proxy server establishes connections, the proxy authenticates them using whatever authentication mechanism is defined in the configuration of the external server. These connections will be re-used for all types of operations to be forwarded to the backend server. The bind DN and password are configured in the directory proxy server.

Whenever a client sends a bind request to the directory proxy server, the server looks at the type of bind request that was sent. If it is a SASL bind request, then the authentication is processed by the directory proxy server itself and it will not be forwarded to the backend server. However, the directory proxy server may use information contained in the backend server as needed. If the bind request is a simple bind request and the bind DN is within the scope of data supplied by the

backend server, then the directory proxy server will forward the client request to the backend server so that it will use the credentials provided by the client.

Regardless of the authentication method that the client uses, the directory proxy server will remember the identity of the client after the authentication is complete and for any subsequent requests sent by that client, it will use the configured authorization method to identify the client to the backend server. Even though the operation is forwarded over a connection that is authenticated as a user defined in the directory proxy server configuration, the request is processed by the backend server under the authority of the end client.

About Client Connection Policies

Client connection policies define the general behavior the server exhibits when communicating with a set of clients. Each policy consists of the following:

- A set of connection criteria that define which client is associated with the policy based on information the server has about the client, including client address, protocol used, secure communication mechanism, location of the client's entry in the directory and the contents of the client's entry. These criteria are the same as those used for filtered logging. For example, different client connection policies could be established for different classes of users, such as root and non-root users.
- A set of constraints on the type of operations a client may request. You can specify whether a particular type of operation is allowed for clients. For some operation types, such as extended operations, you can allow only a particular subset of an operation type, such as a particular extended operation.
- A set of subtree views that define information about the parts of the DIT the client may access.

When a client connection is established, only one client connection policy is applied. If the criteria for several policies match the same client connection, the evaluation order index is used as a tiebreaker. If no policy matches, the client connection is terminated. If the client binds, changing its identity, or uses StartTLS to convert from an insecure connection to a secure connection, then the connection may be evaluated again to determine if it matches the same or a different client connection policy. The connection can also be terminated if it no longer matches any policy.

For information about configuring a client connection policy, see *Configuring Client Connection Policies* on page 77.

About Entry Balancing

Entry balancing allows you to automatically spread entries below a common parent among multiple sets of directory servers for improved scalability and performance. Entry balancing can take advantage of a global index, an in-memory cache used to quickly determine which set or sets of servers should be used to process a request based on the entry DNs and/or the attribute values used in the request.

For information about configuring entry balancing, see *Configuring Entry Balancing*.

Server Component Architecture

This section provides an overview of the process flow between the directory proxy server components, for both a simple proxy deployment and an entry balancing deployment.

Architecture of a Simple Directory Proxy Server Deployment

In a simple directory proxy server deployment, a client request is first processed by a client connection policy as illustrated in Figure 1, “Process Flow for Directory Proxy Server”.

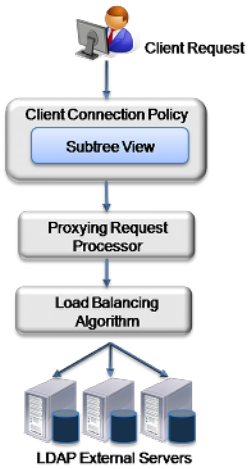


Figure 1: Process Flow for Directory Proxy Server

The client connection policy contains a subtree view, which defines the portion of the DIT available to clients. Once the directory proxy server determines that the DIT is available, it passes the request to the request processor, which defines the logic for processing the request. The request processor then passes the request to a load-balancing algorithm, which determines the server in a set of servers responsible for handling the request. Finally, the request is passed to the LDAP external server. The LDAP external server contains properties that define the server’s location in a topology and the health checks used to determine if the server is functioning properly. This information may be used by the load-balancing algorithm in the course of determining how to route requests.

Architecture of an Entry-Balancing Proxy Server Deployment

Figure 2, “Entry-Balancing Directory Proxy Server Process Flow” describes how a client request is treated in an entry-balancing deployment.

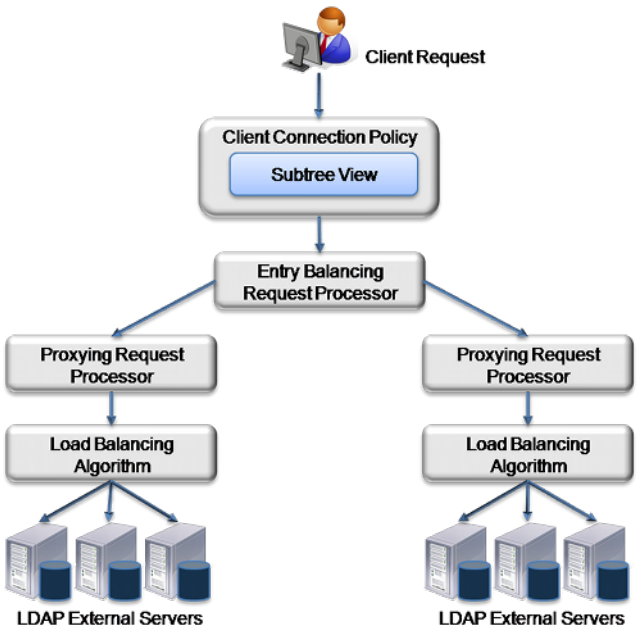


Figure 2: Entry-Balancing Proxy Server Process Flow

Entry balancing is typically used when the data set it too large to fully cache on a single server or when the write performance requirements of an environment are higher than can be achieved with a single replicated set of servers. In such cases, the data may be split across multiple sets of servers, increasing the memory available for caching and the overall write performance in proportion to the number of server sets.

As with a simple proxy deployment, the client request is first processed by the client connection policy, which determines how the directory proxy server communicates with a set of clients. It contains a subtree view that represents the base DN for the entire deployment. The data set splits beneath this base DN.

The request is then passed to the entry-balancing request processor. The entry-balancing request processor contains a global attribute index property, which helps the request processor determine which server set contains the entry and how to properly route the request. It also contains a placement algorithm, which helps it select the server set in which to place new entries created by add requests.

Beneath the entry-balancing request processor are multiple proxying request processors that handle multiple unique sets of data. These request processors pass the request to a load-balancing algorithm, which determines which LDAP external server should handle the request. As with a simple proxy deployment, this LDAP external server contains properties that define the server's location and the health checks used to determine if the server is functioning properly.

Directory Proxy Server Configuration Overview

The configuration of the directory proxy server involves the following steps:

- **Configuring the locations for your deployment**. A location is a collection of servers that share access and latency characteristics. For example, your deployment might include two data centers, one in the east and one in the west. These data centers would be configured as two locations in the directory proxy server. Each location is associated with a name and an ordered list of failover locations, which could be used if none of the servers in the preferred location are available.
- **Configuring the directory proxy server location**. You need to update the configuration to specify the location of the directory proxy server instance.
- **Configuring health checks for the LDAP external servers**. You can configure at what point the proxy server considers an LDAP external server to be available, of degraded availability, or unavailable. Each health check can be configured to be used automatically for all LDAP external servers or for a specified set of servers.
- **Configuring the LDAP external servers**. During this step, you define each of the external directory servers, including the server type. You can configure UnboundID Directory Servers, Alcatel-Lucent 8661 Directory Servers, SunTM Java System Directory Servers, or generic LDAP servers. You also assign the server-specific health checks configured in the previous step.
- **Configuring the load-balancing algorithm**. You configure the load-balancing algorithm used by the directory proxy server to determine which server in a set of similar servers should be used to process a client request. The directory proxy server provides default algorithms. It also steps you through the creation of new algorithms by using an existing algorithm as a template or by creating one from scratch.
- **Configuring the proxying request processor**. In this step, you configure proxying request processors that forward operations received by the directory proxy server to other LDAP external servers.
- **Configuring subtree views**. A subtree view defines the portion of the DIT available to a client. Each subtree view can be associated with a load-balancing algorithm to help distribute the work load.
- **Configuring the client connection policy**. You configure policies to classify how different client connections are managed by the directory proxy server. The client connection policy can be used to control the types of operations that a client may perform and the portion of the DIT that the client can access. Restrictions configured in a client connection policy will take precedence over any capabilities granted by access control or privileges.

Chapter 2 Installing the Directory Proxy Server

This section describes how to install UnboundID® Directory Proxy Server. It includes pre-installation requirements and considerations.

It includes the following sections:

Topics:

Before You Begin

The following sections describe requirements and considerations you should make before installing the software and configuring the UnboundID® Directory Proxy Server objects.

Defining a Naming Strategy

The various objects you will be defining in the UnboundID® Directory Proxy Server will be specific to a particular location or set of servers. Keep this in mind when you are naming objects, so that they will be easy to identify and group with like objects. For example, all of the servers in the west location could be named using a prefix of “west” so that they are readily identifiable when listed. A health check you want to apply only to these servers could also contain the word “west” to make it easier to remember to which group of servers it applies.

Software Requirements: Java

For optimized performance, the UnboundID® Directory Proxy Server requires Java for 32-bit and 64-bit architectures, respectively, depending on your system requirements. You can view the minimum required Java version on your Customer Support Center portal or contact your authorized support provider for the latest software versions supported.

Even if your system already has Java installed, you may want to create a separate Java installation for use by the UnboundID® Directory Proxy Server to ensure that updates to the system-wide Java installation do not inadvertently impact the Directory Proxy Server. This setup requires that the JDK, rather than the JRE, for both the 32-bit and 64-bit versions or both depending on your operating system, be downloaded.

On Solaris systems, if you want to use the 64-bit version of Java, you need to install both the 32-bit and 64-bit versions. The 64-bit version of Java on Solaris is not a full stand-alone installation, but instead relies on a number of files provided by the 32-bit installation. Therefore, the 32-bit version should be installed first, and then the 64-bit version installed in the same location with the necessary additional files.

On other platforms (for example, Linux and Microsoft Windows), the 32-bit and 64-bit versions of Java contain complete installations. If you only want to run the 64-bit version of Java, then it is not necessary to install the 32-bit JDK. If you want to have both versions installed, then they should be installed in separate directories, because the files cannot co-exist in the same directory as they can on Solaris systems.

To Install Java (Oracle/Sun)

1. Open a browser and navigate to the following Oracle download site:

```
http://www.oracle.com/technetwork/java/javase/downloads/index.html
```

2. Download the latest version Java JDK. Click the JDK Download button corresponding to the latest Java update.

3. On the Java JDK page, click the Accept Licence Agreement button, then download the version based on your operating system.

To Install Java (IBM)

1. Open a browser and navigate to the following IBM download site:

```
http://www.ibm.com/developerworks/java/jdk/
```

2. Select the Java version for your operating system.

Preparing the Operating System

You should make the following changes to your operating system depending on the production environments on which the UnboundID® Directory Proxy Server will run.

To Set the File Descriptor Limit (Linux)

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

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

```
ulimit -aH
```

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

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

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

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

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

```
# ulimit -n
```

Setting the Filesystem Flushes

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

```
commit=1
```

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

About Editing OS-Level Environment Variables

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

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

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

Install sysstat and pstack (Red Hat)

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

Getting the Installation Packages

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

To Unpack the Build Distribution

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

```
$ unzip UnboundID-Proxy-<version>.zip
```

You can now set up the Directory Proxy Server.

Installing the Directory Proxy Server

When you deploy UnboundID® Directory Proxy Server in a topology, you generally deploy them in pairs. These pairs are configured identically except for their host name, port name, and possibly their location.

To help administrators easily install identical proxies, the directory proxy server allows you to clone a proxy configuration. First, you install a directory proxy server using the `setup` tool. Then, you configure it using the `create-initial-proxy-config` tool described in *Using the create-initial-proxy-config Tool*. Finally, you run the `setup` tool on subsequent servers, indicating that you want to clone the configuration on a peer server.

The following sections describe the setup tool in more detail, and tell you how to install first and subsequent directory proxy servers in your directory topology.

About the setup Tool

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

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

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

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

Installing the First Directory Proxy Server in Interactive Mode

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

To Install the First Directory Proxy Server in Interactive Mode

1. Change to the server root directory.

```
$ cd UnboundID-Proxy
```

2. Use the `setup` command to install the Directory Proxy Server instance from the server root directory.

```
$ ./setup
```

Note: If your *JAVA_HOME* environment variable is set to an older version of Java, you must explicitly specify the path to the Java JDK installation during setup. You can either set the *JAVA_HOME* environment variable with the Java JDK path or execute the `setup` command in a modified Java environment using the `env` command.

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

3. Read the UnboundID End-User License Agreement. If you agree to its terms, 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 root user DN, or press **Enter** to accept the default (`cn=Directory Manager`), and then type and confirm the root user password.

6. Enter the port number of your Directory Proxy Server, or press **Enter** to accept the default port, which is 389.

```
On which port would you like the Directory Proxy Server to accept connections from
 LDAP clients? [389]:
```

7. For SSL and StartTLS, type `yes` to enable one or both. Otherwise, press **Enter** to accept the default (no). If you enable SSL or StartTLS, the `setup` tool creates a Directory Proxy Server trust store in the `config/truststore` directory. The pin is encoded in the `config/keystore.pin` file.

```
Do you want to enable SSL? (yes / no) [no]:
Do you want to enable StartTLS? (yes / no) [no]:
```

8. If you answered `yes` for SSL or StartTLS, you will be prompted for the certificate options. If you use the Java or the PKCS#12 key store, you will be asked for the Key Store path, and the key PIN. If you use the PKCS#11 token, you will be asked for only the key PIN.

```
Do you want to enable SSL? (yes / no) [no]: yes
On which port would you like the Directory Proxy Server to accept connections from
 LDAPS clients? [1636]:
```

```
Do you want to enable StartTLS? (yes / no) [no]: yes

Certificate server options:
1) Generate self-signed certificate (recommended for testing purposes only)
2) Use an existing certificate located on a Java Key Store (JKS)
3) Use an existing certificate located on a PKCS#12 key store
4) Use an existing certificate on a PKCS#11 token

Enter choice [1]: 2
Java Key Store (JKS) path: /path/to/keystore
Key store PIN:
```

9. By default, the Directory Proxy Server listens on all available network interfaces for client connections. If this is acceptable, you can skip this step. If you want to limit the client connections to specific host names or IP addresses, type `yes` at the prompt, and then, enter the host name or IP address. You will be prompted again to enter another host name or IP address. Enter as many as applicable. When you are done, press **Enter** to continue. Otherwise, accept the default of `no`.

```
By default the server listens on all available network
interfaces for client connections. Would you like to specify
particular addresses on which this server will listen for
client connections? (yes / no)(no):
```

10. If you want to configure your Directory Proxy Server to use entry balancing, where leaf and non-leaf entries below a common parent entry are distributed among multiple directory servers, type `yes`. Otherwise, accept the default of `no`.

```
Do you anticipate configuring this Proxy Server for entry balancing.
Choosing 'yes' will allow you to specify that more memory be allocated to
the server and tools? (yes / no) [no]: no
```

Note: Answering yes to entry balancing allows more memory to be reserved for the Directory Proxy Server. If you do not specify entry balancing during setup, you will not see any more prompts about entry balancing for the remainder of setup and configuration.

11. If you want to configure an entry balanced proxy server topology, enter yes. Otherwise, aceept the default of no.

12. Next, type `yes` if you want to allocate the amount of memory to the JVM heap for maximized performance. This option should only be selected if the Directory Proxy Server is the primary application and no other processes consume a significant amount of memory.

```
Do you want to tune the JVM of this system such that
the memory dedicated to the server is maximized? Choosing 'yes'
will allow you to optionally specify the maximum amount of memory
to be allocated to the server and tools (yes / no) [no]:
```

13. If you choose to tune the JVM, enter the maximum amount of memory you want the Directory Proxy Server to allocate to the server and tool. In this example, the maximum allowed for the server is 1 gigabytes.

    The command line provides a dynamic value range based on the resources of the system on which the installer is running. In the example above, the range is 64 megabytes to 16 gigabytes.

```
Enter the maximum amount of memory to be allocated to the
```

```
server and tools. The format for this value is the same as the
-Xmx JVM option which is a number followed by a unit m or g. For example
'2g' means 2 gigabytes. The value must be between '64m' and '16g' [16g]:
```

14. Type `yes`, or press **Enter** to accept the default to start the server after the configuration has completed. If you plan to configure additional settings or import data, you can type `no` to keep the server in shutdown mode.

```
Do you want to start the server when the configuration is completed? (yes /no) [yes]:
```

15. On the **Setup Summary** page, confirm the configuration, and press **Enter** to set up the server. The configuration is recorded in the `/server-root/logs/tools/setup.log` file.

```
Setup Summary
=============
LDAP Listener Port: 389
LDAP Secure Access: disabled
Root User DN:       cn=Directory Manager

Start Server when the configuration is completed

What would you like to do?
      1) Set up the server with the parameters above
      2) Provide the setup parameters again
      3) Cancel the setup

Enter choice [1]:

Configuring Directory Proxy Server..... Done.
Starting Directory Proxy Server....... Done.

See /UnboundID-Proxy/logs/setup.log for a detailed log of this operation.
```

16. Once setup is complete, you are prompted to begin configuration. Select whether you want to create an initial basic configuration using the `create-initial-proxy-config` tool, whether you want to configure by hand using `dsconfig`, or whether to quit and configure your proxy server later. You need to configure your proxy later if you plan to use custom schema in your deployment. For more information about configuring your directory proxy server using this tool, see *Using the create-initial-proxy-config Tool*.

```
This server is now ready for configuration What would you like to do?
   1) Start 'create-initial-proxy-config' to create a basic
      initial configuration (recommended for new users)
   2) Start 'dsconfig' to create a configuration from scratch
   3) Quit

Enter choice [1]:
```

To Install Additional Directory Proxy Servers in Interactive Mode

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

1. Change to the server root directory.

```
$ cd UnboundID-Proxy
```

2. Use the `setup` command to install the Directory Proxy Server instance from the server root directory.

```
$ ./setup
```

Note: If your *JAVA_HOME* environment variable is set to an older version of Java, you must explicitly specify the path to the Java JDK installation during setup. You can either set the *JAVA_HOME* environment variable with the Java JDK path or execute the `setup` command in a modified Java environment using the `env` command.

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

3. Read the UnboundID End-User License Agreement. If you agree to its terms, type `yes` to continue.

4. Enter `yes` in response to add this new server to an existing topology.

```
Would you like to add this server to an existing Directory Proxy Server topology?
 (yes / no) [no]: yes
```

5. Enter the host name of the directory proxy server from which configuration settings are copied during setup.

```
Enter the hostname of the peer Directory Proxy Server from which you would like
to copy configuration settings. [proxy.example.com]:
```

6. Type the port number of the peer proxy server from which configuration settings are copied during setup. You can press **Enter** to accept the default port, which is 389.

```
Enter the port of the peer Directory Proxy Server [389]:
```

7. Enter the option corresponding to the type of connection you want to use to connect to the peer Directory Proxy Server.

```
How would you like to connect to the peer Directory Proxy Server?
  1) None
  2) SSL
  3) StartTLS

Enter choice [1]:
```

8. Type the root user DN of the peer directory proxy server, or press **Enter** to accept the default (cn=Directory Manager), and then type and confirm the root user password.

```
Enter the manager account DN for the peer Directory Proxy Server [cn=Directory
 Manager]:
Enter the password for cn=Directory Manager:
```

9. Enter the host name of the new local directory proxy server.

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

10. Choose the location of your new directory proxy server instance or enter a new one.

```
Choose the location for this Directory Proxy Server
  1) east
  2) Specify a new location

Enter choice [1]:
```

11. Enter the root user DN, or press **Enter** to accept the default (`cn=Directory Manager`), and then type and confirm the root user password.

12. Enter the port number of your Directory Proxy Server, or press **Enter** to accept the default port, which is 389.

```
On which port would you like the Directory Proxy Server to accept connections from
 LDAP clients? [389]:
```

13. For SSL and StartTLS, type `yes` to enable one or both. Otherwise, press **Enter** to accept the default (no). If you enable SSL or StartTLS, the `setup` tool creates a Directory Proxy Server trust store in the `config/truststore` directory. The pin is encoded in the `config/keystore.pin` file.

```
Do you want to enable SSL? (yes / no) [no]:
Do you want to enable StartTLS? (yes / no) [no]:
```

14. If you answered `yes` for SSL or StartTLS, you will be prompted for the certificate options. If you use the Java or the PKCS#12 key store, you will be asked for the Key Store path, and the key PIN. If you use the PKCS#11 token, you will be asked for only the key PIN.

```
Do you want to enable SSL? (yes / no) [no]: yes
On which port would you like the Directory Proxy Server to accept connections from
 LDAPS clients? [1636]:
Do you want to enable StartTLS? (yes / no) [no]: yes

Certificate server options:
1) Generate self-signed certificate (recommended for testing purposes only)
2) Use an existing certificate located on a Java Key Store (JKS)
3) Use an existing certificate located on a PKCS#12 key store
4) Use an existing certificate on a PKCS#11 token

Enter choice [1]: 2
Java Key Store (JKS) path: /path/to/keystore
Key store PIN:
```

15. By default, the Directory Proxy Server listens on all available network interfaces for client connections. If this is acceptable, you can skip this step. If you want to limit the client connections to specific host names or IP addresses, type `yes` at the prompt, and then, enter the host name or IP address. You will be prompted again to enter another host name or IP address. Enter as many as applicable. When you are done, press **Enter** to continue. Otherwise, accept the default of `no`.

```
By default the server listens on all available network
interfaces for client connections. Would you like to specify
particular addresses on which this server will listen for
client connections? (yes / no)(no):
```

Installing the First Directory Proxy Server in Non-Interactive Mode

You can run the `setup` command in non-interactive mode to automate the installation process using a script or to run the command directly from the command line. If there is a missing or incorrect argument, the `setup` tool fails and aborts the process.

The `setup` tool automatically chooses the maximum heap size. You can manually tune the maximum amount of memory devoted to the server's process heap using the `--maxHeapSize`

option. The `--maxHeapSize` argument is only valid if the `--entryBalancing` or `--aggressiveJVMTuning` options are also present.

For example, use the `--aggressiveJVMTuning` option to set the maximum amount of memory used by the directory proxy server and tools as follows:

```
--aggressiveJVMTuning --maxHeapSize 256m
```

If you are using entry balancing, tune the amount of memory devoted to the directory proxy server using the `--entryBalancing` option as follows:

```
--entryBalancing --maxHeapSize 1g
```

The amount of memory allowed when using the `--entryBalancing` option is calculated and depends on the amount of system memory available. If you are using entry balancing and also want the tools to get more memory, include both the `--entryBalancing` and the `--aggressiveJVMTuning` options.

```
--entryBalancing --aggressiveJVMTuning --maxHeapSize 1g
```

If you have already configured a trust store, you can also use the setup tool to enable security. The following example enables security, both SSL and StartTLS. It also specifies a JKS keystore and truststore that define the server certificate and trusted CA. The passwords for the keystore files are defined in the corresponding .pin files, where the password is written on the first line of the file. The values in the .pin files will be copied to the server-root/config directory in the keystore.pin file.

Note that the password to the private key within the key store is expected to be the same as the password to the key store. If this is not the case, the private key password can be defined within the Directory Proxy Management Console or `dsconfig` by editing the Trust Manager Provider standard configuration object.

```
$ env JAVA_HOME=/ds/java ./setup --cli \
   --no-prompt --rootUserDN "cn=Directory Manager" \
   --rootUserPassword "password" --ldapPort 389 \
   --enableStartTLS --ldapsPort 636 \
   --useJavaKeystore /path/to/devkeystore.jks \
   --keyStorePasswordFile /path/to/devkeystore.pin \
   --certNickName server-cert \
   --useJavaTrustStore /path/to/devtruststore.jks \
   --trustStorePasswordFile /path/to/devtruststore.pin \
   --acceptLicense
```

To Install the First Directory Proxy Server in Non-Interactive Mode

- Use `setup` with the `--no-prompt` option. The command uses the default root user DN (`cn=Director Manager`) with the specified `--rootUserPassword` option. You must include the `--acceptLicense` option or the `setup` tool will generate an error message.

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

To Install Additional Directory Proxy Server in Non-Interactive Mode

You can run the `setup` command in non-interactive mode to automate the installation process using a script or to run the command directly from the command line. If there is a missing or incorrect argument, the `setup` tool fails and aborts the process.

To Install Additional Directory Proxy Server in Non-Interactive Mode

- Use `setup` with the `--no-prompt` option.

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

Installing the Directory Proxy Server with a Truststore in Non-Interactive Mode

If you have already configured a trust store, you can also use the `setup` tool to enable security. The following example enables SSL security. It also specifies a JKS keystore and truststore that define the server certificate and trusted CA. The passwords for the keystore files are defined in the corresponding .pin files, where the password is written on the first line of the file. The values in the .pin files will be copied to the `server-root/config` directory in the `keystore.pin` and `truststore.pin` files.

Note: The password to the private key within the key store is expected to be the same as the password to the key store. If this is not the case, the private key password can be defined within the Directory Proxy Management Console or `dsconfig` by editing the Key Manager Provider standard configuration object.

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

- Run the `setup` tool to install a directory proxy server with a truststore.

```
$ env JAVA_HOME=/ds/java ./setup --cli \
  --no-prompt --rootUserDN "cn=Directory Manager" \
  --rootUserPassword "password" \
  --ldapPort 389 --ldapsPort 636 \
  --useJavaKeystore /path/to/devkeystore.jks \
  --keyStorePasswordFile /path/to/devkeystore.pin \
  --certNickName server-cert \
  --useJavaTrustStore /path/to/devtruststore.jks \
  --acceptLicense

In order to update the trust store, the password must be provided

See 'prepare-external-server --help' for general overview

Testing connection to ds-east-01.example.com:1636 ..... Done
Testing 'cn=Proxy User,cn=Root DNs,cn=config' access .....
Created 'cn=Proxy User,cn=Root DNs,cn=config'
```

```
Testing 'cn=Proxy User,cn=Root DNs,cn=config' access ..... Done
Testing 'cn=Proxy User,cn=Root DNs,cn=config' privileges ..... Done
Verifying backend 'dc=example,dc=com' ..... Done
```

Running the Server

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

To Start the Directory Proxy Server

Use bin/start-proxy to start the server.

```
$ bin/start-proxy
```

To Run the Server as a Foreground Process

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

   ```
   $ bin/start-proxy --nodetach
   ```

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

To Start the Server at Boot Time

By default, the UnboundID® Directory Proxy Server does not start automatically when the system is booted. Instead, you must manually start it with the bin/start-proxy command. To configure the Directory Proxy Server to start automatically when the system boots, use the `create-rc-script` utility to create a run control (RC) script, or create the script manually.

1. Create the startup script.

   ```
   $ bin/create-rc-script --outputFile UnboundID-Proxy.sh --userName ds
   ```

2. As a root user, move the generated UnboundID-Proxy.sh script into the `/etc/init.d` directory and create symlinks to it from the `/etc/rc3.d` directory (staring with an “S” to ensure that the server is started) and `/etc/rc0.d` directory (starting with a “K” to ensure that the server is stopped).

   ```
   # mv UnboundID-Proxy.sh /etc/init.d/
   # ln -s /etc/init.d/UnboundID-Proxy.sh/etc/rc3.d/S50-boot-ds.sh
   # ln -s /etc/init.d/UnboundID-Proxy.sh /etc/rc0.d/K50-boot-ds.sh
   ```

Some Linux implementations may not like the “-” in the scripts. If your scripts do not work, try renaming the scripts without the dashes. You can also try symlinking the S50* file into the `/etc/rc3.d` or the `/etc/rc0.d` directory or both, based on whatever runlevel the server enters when it starts. Some Linux systems do not even use `init.d-style` startup scripts, so depending on whatever flavor of Linux you are using you might have to put the script somewhere else or use some other mechanism for having it launched at startup.

3. Log out as root, and re-assume the ds role if you are on a Solaris system.

Stopping the Directory Server

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

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

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

To Stop the Server

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

```
$ bin/stop-proxy
```

To Schedule a Server Shutdown

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

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

```
$ bin/stop-ds --stopTime 20120606134500Z --port 1389 \
  --bindDN "uid=admin,dc=example,dc=com" --bindPassword secret \
  --stopReason "Scheduled offline maintenance"
```

To Restart the Server

You can re-start the Directory Proxy Server using the bin/stop-proxy command with the `--restart` or `-R` option. Running the command is equivalent to shutting down the server, exiting the JVM session, and then starting up again. Shutting down and restarting the JVM requires a re-priming of the JVM cache. To avoid destroying and re-creating the JVM, use an in-core restart, which can be issued over LDAP. The in-core restart will keep the same Java process and avoid any changes to the JVM options.

- Go to the installation directory. Using an in-core restart (via the loopback interface), run the bin/stop-proxy command with the `-R` or `--restart` options.

```
$ bin/stop-proxy --restart --hostname 127.0.0.1 --port 1389 \
  --bindDN “uid=admin,dc=example,dc=com” --bindPassword secret
```

About the Server Uninstallation Modes

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

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

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

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

```
$ rm -rf /ds/UnboundID-Proxy
```

To Uninstall the Server in Interactive Mode

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

1. From the installation directory, run the `uninstall` command.

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

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

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

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

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

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

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

4. If the Directory Proxy Server is part of a replication topology, type `yes` to provide your authentication credentials (Global Administrator ID and password). If you are uninstalling a stand-alone server, continue to step 7.

5. Type the Global Administrator ID and password to remove the references to this server in other replicated servers. Then, type or verify the host name or IP address for the server that you are uninstalling.

6. Next, select how you want to trust the server certificate if you have set up SSL or StartTLS. For this example, press **Enter** to accept the default.

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

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

Enter choice [3]:
```

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

To Uninstall the Server in Non-Interactive Mode

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

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

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

2. If any files or directories remain, manually remove them.

To Uninstall Selected Components in Non-Interactive Mode

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

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

Updating the Directory Proxy Server

UnboundID issues new software builds periodically and distributes the software package in zip format. Administrators can use the directory proxy server's `update` utility to update the current server code with the latest features and bug fixes. To update the directory proxy server to a newer version, download the build package, and then unzip the new server package on the same host as the server that you wish to update. Before upgrading a server, you should ensure that it is capable of starting without severe or fatal errors.

During an update process, the updater checks a manifest file that contains a MD5 checksum of each file in its original state when installed from zip. Next, it compares the checksum of the new server files to that of the old server. Any files that have different checksums will be updated. For files that predates the manifest file generation, the file is backed up and replaced. The updater also logs all file changes in the history directory to tell what files have been changed.

For schema updates, the `update` tool preserves any custom schema definitions (`99-user.ldif`). For any default schema element changes, if any, the updater will warn the user about this condition and then create a patch schema file and copy it into the server's schema directory. For configuration files, the update tool preserves the configuration file, `config.ldif`, unless new configuration options must be added to the directory proxy server.

Once the updater finishes its processing, it checks if the newly updated server starts without any fatal errors. If an error occurs during the update process, the `update` tool reverts the server root instance to the server state prior to the update.

To Update the Directory Proxy Server

Assume that an existing version of the directory proxy server is stored at UnboundID-Proxy-old, which you want to update.

1. Make sure you have complete, readable backup of the existing system before upgrading the directory proxy server build. Also, make sure you have a clear backout plan and schedule.

2. Download the latest version of the UnboundID® Directory Proxy Server software and unzip the file. For this example, let's assume the new server is located in the UnboundID-Proxy-new directory.

3. Check the version number of the newly downloaded directory proxy server instance using the `--version` option on any command-line utility. For example, you should see the latest revision number.

```
$ UnboundID-Proxy-new/setup --version UnboundID® Directory Proxy Server 3.6.0.0
Build 2011043200609Z Revision 9235
```

4. Use the `update` tool of the newly unzipped build to update the directory proxy server code. Make sure to specify the directory proxy server instance that you are upgrading with the `--serverRoot` option. The directory proxy server must be stopped for this update to be applied.

```
$ UnboundID-Proxy-new/update --serverRoot UnboundID-Proxy-old
```

Note: The UnboundID® Directory Proxy Server provides a web console called the Directory Proxy Management Console, to configure and monitor the server. If you update the directory proxy server version, you should also update the Directory Proxy Management Console.

5. View the log file to see which files were changed. The log file is located in the `<server-root>/history` directory. For example, the file will be labelled with the directory proxy server version number and revision.

```
$ view <server-root>/history/1272307020420-3.6.0.0.9235/update.log
```

Reverting an Update

Once the directory proxy server has been updated, you can revert to the most recent version (one level back) using the `revert-update` tool. The `revert-update` tool accesses a log of file actions taken by the updater to put the filesystem back to its prior state. If you have run multiple updates, you can run the `revert-update` tool multiple times to revert to each prior update sequentially. You can only revert back one level. For example, if you have run the update twice since first installing the directory proxy server, you can run the `revert-update` command to revert to its previous state, then run the `revert-update` command again to return to its original state.

Note: The UnboundID® Directory Proxy Server will be stopped during the `revert-update` process.

To Revert to the Most Recent Server Version

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

```
$ UnboundID-Proxy-old/revert-update
```

Installing the Directory Proxy Management Console

The UnboundID® Directory Proxy Server provides a graphical web application tool, the UnboundID Directory Proxy Management Console. The Directory Proxy Management Console provides configuration and schema management functionality in addition to monitoring and server information. Like the `dsconfig` configuration tool, all changes made using the Directory Proxy Management Console are recorded in `logs/config-audit.log`. In addition, anytime a configuration is made to the system, the configuration backend is automatically updated and saved as gzip-compressed files. You can access the changes in the `config/archived-configs` folder.

The Directory Proxy Management Console is a web application that must be deployed in a servlet container that supports the servlet API 2.5 or later. An installation using Apache Tomcat is described below for illustration purposes only.

Note: The Directory Proxy Management Console supports JBoss 7.1.1 or later. Refer to the JBoss Compatibility section in the `WEB-INF/web.xml` file for specific configuration steps.

To Install the Directory Proxy Management Console Out of the Box

1. Download and install the servlet container. For example, download `apache-tomcat-<version>.zip` from http://tomcat.apache.org/, and then unzip this file in a location of your choice.

   Note:

2. Set the appropriate Apache Tomcat environment variables. The `setclasspath.sh` and `catalina.sh` files are in the tomcat `bin` directory.

   ```
   $ echo "BASEDIR=/path/to/tomcat" >> setclasspath.sh
   $ echo "CATALINA_HOME=/path/to/tomcat" >> catalina.sh
   ```

3. Download the Directory Proxy Management Console ZIP file, `UnboundID-Proxy-web-console-<version>.zip` and unzip the file on your local host. You should see the following files:

```
3RD-PARTY-LICENSE.TXT
LICENSE.TXT
README
dsconsole.war
```

4. Create a `proxyconsole` directory in `apache-tomcat-<version>/webapps/proxyconsole`. Then, copy the `dsconsole.war` file to `apache-tomcat-<version>/webapps/proxyconsole`.

```
$ mkdir apache-tomcat-<version>/webapps/proxyconsole
$ cp dsconsole.war apache-tomcat-<version>/webapps/proxyconsole
```

5. Go to the `apache-tomcat-<version>/webapps/proxyconsole` directory to extract the contents of the console. The jar command is included with the JDK.

```
$ cd apache-tomcat-<version>/webapps/proxyconsole
$ jar xvf proxyconsole.war.war
```

6. Optional. Edit the `WEB-INF/web.xml` file to point to the correct Directory Proxy Server instance. Change the host and port to match your server. The parameters in the `web.xml` file appear between <!-- and --> as comments. Uncomment the parameters you need to use. For example, you can specify the server or servers that the console uses to authenticate using the following parameters:

```
<context-param>
   <param-name>ldap-servers</param-name>
   <param-value>localhost:389</param-value>
</context-param>
```

Note: If the `ldap-servers` parameter is left as-is (i.e., undefined by default), the web console displays a form field for the user to enter the server host and port.

7. Optional. With the default configuration, Tomcat will time out sessions after 30 minutes of inactivity, forcing the user to log back in again. This can be changed on a servlet container wide basis by editing `apache-tomcat-<version>/conf/web.xml`, and updating the value of this configuration parameter:

```
<session-config>
     <session-timeout>120</session-timeout>
</session-config>
```

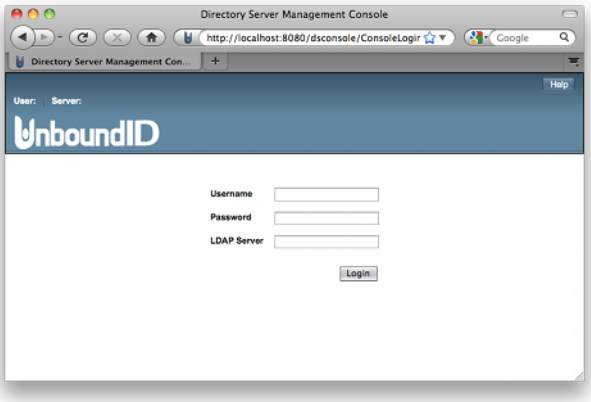
   The session expires after the specified number of minutes. Changing the value to 120, for example, will extend the expiration to two hours. Changes to this setting might not take effect until the servlet container is restarted, so consider changing the value before starting the server for the first time.

8. Start the Directory Proxy Server if it is not already running, and then start the Directory Proxy Management Console using the `apache-tomcat-<version>/bin/startup.sh` script. Use `shutdown.sh` to stop the servlet container. (On Microsoft Windows, use `startup.bat` and `shutdown.bat`.) Note that the *JAVA_HOME* environment variable must be set to specify the location of the Java installation to run the server.

```
$ env JAVA_HOME=/ds/java bin/startup.sh
Using CATALINA_BASE:    /apache-tomcat-<version>
Using CATALINA_HOME:    /apache-tomcat-<version>
Using CATALINA_TMPDIR:  /apache-tomcat-<version>/temp
```

```
Using JRE_HOME:         /ds/java
```

9. Open a browser to `http://hostname:8080/proxyconsole`. By default, Tomcat listens on port 8080 for HTTP requests.



Note: If you re-start the Directory Proxy Server, you must also log out of the current Directory Proxy Management Console session and then log back in to start a new console session.

Working with the Directory Proxy Management Console

The Directory Proxy Management Console does not persistently store any credentials for authenticating to the directory proxy server but uses the credentials provided by the user when logging in. When managing multiple server instances, the provided credentials must be valid for each instance.

Note: When managing multiple servers, the admin user must be created on each of the managed users. The `cn=admin data` entry is not replicated in the directory proxy server.

Logging into the Directory Proxy Management Console

To log into the console, you can either use a DN (for example, cn=Directory Manager) or provide the name of an administrator, which is stored under cn=admin data. The `dsframework` command can be used to create a global administrator, for example:

```
$ dsframework create-admin-user \
  --hostname server1.example.com \
  --port 1389 --bindDN "cn=Directory Manager" \
  --bindPassword secret \
  --userID someAdmin --set password:secret
```

To Log into the Directory Proxy Management Console

1. Go to the installation directory.

```
$ cd UnboundID-Proxy
```

2. Start the Directory Proxy Server.

```
$ bin/start-proxy
```

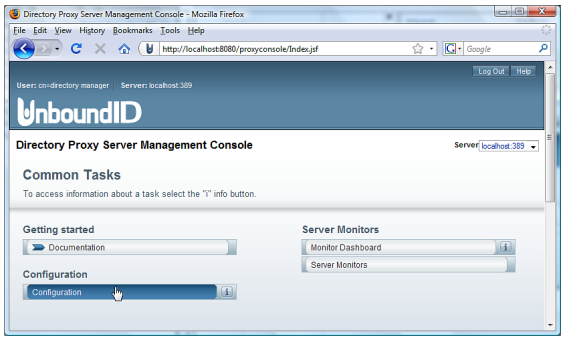
3. Start the Apache Tomcat application server.

```
$ /apache-tomcat-<version>/bin/startup.sh
```

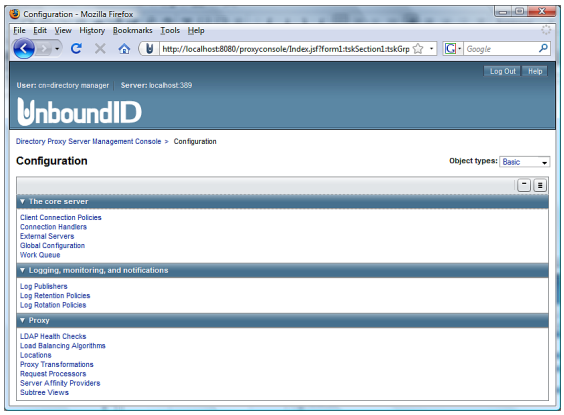
4. Open a browser to http://hostname:8080/proxyconsole/.

5. Type the root user DN (or any authorized administrator user name) and password, and then click **Login**.

6. On the Directory Proxy Management Console, click **Configuration**.



7. View the Configuration menu. By default, the console displays the Basic object type properties. You can change the complexity level of the object types using the **Object Types** drop-down list.



To Upgrade the Directory Proxy Management Console

1. Shut down the console and servlet container.
2. In the current deployment of the Directory Proxy Management Console, move the `webapps/proxyconsole/WEB-INF/web.xml` file to another location.
3. Download and deploy the latest version for the Directory Proxy Management Console. Follow steps 2–5 outlined in the section "To Install the Console Out of the Box".
4. Assuming you had not renamed the `.war` file when you originally deployed the Directory Proxy Management Console, run a diff between the previous and newer version of the web.xml file to determine any changes that should be applied to the new web.xml file. Make those changes to the new file, and then replace the newly deployed Directory Proxy Management Console's `web.xml` to `webapps/proxyconsole/WEB-INF/web.xml`.
5. Start the servlet container.

To Uninstall the Directory Proxy Management Console

1. Close the Directory Proxy Management Console, and shut down the servlet container. (On Microsoft Windows, use `shutdown.bat`).

```
$ apache-tomcat-<version>/bin/shutdown.sh
```

2. Remove the `webapps/proxyconsole` directory.

```
$ rm -rf webapps/proxyconsole
```

3. Restart the servlet container instance if necessary. Alternatively, if no other applications are installed in the servlet instance, then the entire servlet installation can be removed by deleting the servlet container directory.

Chapter 3 Configuring the Directory Proxy Server

Once you have initially configured the UnboundID® Directory Proxy Server, you can manage your deployment using the configuration framework and management tools. This chapter briefly describes these tools and provides procedures to help you maintain and update your deployment.

It includes the following sections:

Topics:

Overview of the Configuration and Management Tools

The UnboundID® Directory Proxy Server configuration can be accessed and modified in the following ways:

- **The `create-initial-proxy-config` tool**. This command-line tool can be used to initially configure the directory proxy server. We strongly recommend that you use the `create-initial-proxy-config` tool for your initial directory proxy server configuration. This tool prompts you for basic information about your topology, including external servers, their locations, and credentials for communicating with them. Once the configuration is complete, the tool writes the configuration to a `dsconfig` batch file and allows you to apply the configuration to the local directory proxy server.
- **Using the Directory Proxy Management Console**. The UnboundID® Directory Proxy Server provides a web-based console for graphical server management and monitoring. The console provides equivalent functionality as the `dsconfig` command for viewing or editing configurations. All configuration changes using this tool are recorded in `logs/config-audit.log`, which also has the equivalent reversion commands should you need to back out of a configuration.
- **Using the dsconfig Command-Line Tool**. The `dsconfig` tool is a text-based menu-driven interface to the underlying configuration. The tool runs the configuration using three operational modes: interactive command-line mode, non-interactive command-line mode, and batch mode. All configuration changes made using this tool are recorded in `logs/config-audit.log`.
- **The `prepare-external-server` tool**. This tool can be used to configure communication between the UnboundID® Directory Proxy Server and the external directory server. This tool can be used in conjunction with `create-initial-proxy-config` or dsconfig to simplify configuring your directory proxy server deployment. For more information about using this tool, see *To Configure Server Communications Using the prepare-external-server Tool*.

Using the create-initial-proxy-config Tool

The `create-initial-proxy-config` tool helps you to initially configure the local directory proxy server. You are prompted to launch this tool after installing the directory proxy server. The tool assumes the following about your topology:

- All servers are accessible through a single user account. This user account must be a root user that is not generally accessible to clients to avoid inadvertent changes, deletions, or backend server availability issues due to reimporting data.
- All servers support the same type of communication security.
- All external servers are any combination of UnboundID Directory Server, Alcatel-Lucent 8661 Directory Server, Sun Directory Server, or Red Hat (including Fedora and 389) instances.

If your topology does have these characteristics, you can use the tool to define a basic configuration that is saved to a `dsconfig` batch file. You can then run the `dsconfig` tool to fine-tune the configuration. You can also use this tool to configure an entry balancing configuration, which allows you to automatically spread entries below a common parent among multiple sets of directory servers for improved scalability and performance.

The `create-initial-proxy-config` tool produces a log file called `create-initial-proxy-config.log` that is stored in the local directory proxy server's logs directory.

You can only run the `create-initial-proxy-config` tool once for the initial configuration of each directory proxy server instance. To tune your configuration, use the `dsconfig` tool. When installing a second directory proxy server, it will not be necessary to run the `create-initial-proxy-config` tool again, as the directory proxy server setup has the ability to clone the settings from an existing proxy server.

This section describes how to use this tool to configure a standard directory proxy server deployment as well as an entry balancing configuration.

Configuring a Standard Proxy Server Deployment

This section describes how to install a standard directory proxy server deployment using the `create-initial-proxy-config` tool. Remember that you deploy directory proxy servers in pairs. Each pair should be configured identically except for their host name, port, and possibly their location.

To Configure a Standard Directory Proxy Server Deployment

1. After initial installation, select the number to start the `create-initial-proxy-config` tool automatically. Otherwise, run it manually at the command line from the installation directory, <server-root>/UnboundID-Proxy.

   ```
   $ ./bin/create-initial-proxy-config
   ```

2. Press return to continue with configuration.

   ```
   Would you like to continue? (yes / no) [yes]:
   ```

3. Enter the DN for the directory proxy user account, then enter and confirm the password for this account. Note that you should not use `cn=Directory Manager` as the account to use for communication between the directory proxy server and the directory server. For security reasons, the account used to communicate between the directory proxy server and the directory server should not be directly accessible by clients accessing the directory proxy server. For more information about this account, see *Configuring LDAP External Servers*.

   ```
   Enter the DN of the proxy user account [cn=Proxy User,cn=Root DNs,cn=config]:
   Enter the password for 'cn=Proxy User,cn=Root DNs,cn=config':
   Confirm the password for 'cn=Proxy User,cn=Root DNs,cn=config':
   ```

4. Specify whether you will be using secure communication with the directory server instances.

```
  1)  None
  2)  SSL
  3)  StartTLS

  b)  back
  q)  quit

Enter choice [1]:
```

5. Specify the base DNs of the proxy. You can specify more than one. Press **Enter** when you have finished specifying DNs.

```
Enter a DN or choose a menu item [dc=example,dc=com]:
```

6. Define a location for your server. You can specify multiple locations. This example illustrates defining a location named `east`.

```
Enter a location name or choose a menu item: east
```

7. If you defined more than one location, specify the location that contains the directory proxy server itself.

```
Choose the location for this Directory Proxy Server
  1)  east
  2)  west

  b)  back
  q)  quit

  Enter choice [1]: 1
```

8. Define the host, port, and authentication information used by the LDAP external servers. If you have specified more than one location, you will go through this process for each location.

```
Enter a host:port or choose a menu item [localhost:389]: ldap-east-01.example.com:389
```

9. Select 3 to indicate that you want the tool to create a proxy user account on all of your LDAP external servers.

```
Would you like to prepare ldap-east-01.example.com:389 for access by
the Proxy Server?

    1) Yes
    2) No
    3) Yes, and all subsequent servers
    4) No, and all subsequent servers

Enter choice [1]: 3

Testing connection to ldap-east-01.example.com:389..... Done
Testing 'cn=Proxy User' access to ldap-east-01.example.com:389..... Failed to bind as
 'cn=Proxy User'
```

10. If the proxy user account did not previously exist on your LDAP external server, create the account by connecting as `cn=Directory Manager`.

```
Would you like to create or modify root user 'cn=Proxy User' so that it is available
 for this 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
```

11. Press **Enter** to finish configuring the location.

```
Enter a host:port or choose a menu item [Press ENTER when finished entering servers
 for 'east']:
```

12. Review the configuration summary. Once you have confirmed that the changes are correct, press **Enter** to write the configuration.

```
   b)  back
   q)  quit
   w)  write configuration

   Enter choice [w]:
```

13. Press **Enter** to apply the changes to the directory proxy server. Alternatively, you can quit and instead run the `dsconfig` batch file at a later time. Once the changes have been applied, you cannot use the `create-initial-proxy-config` tool to configure this directory proxy server again. Instead, use the `dsconfig` tool.

```
Apply these configuration changes to the local Directory Proxy Server? (yes /no)
 [yes]:
```

    If you open the generated `dps-cfg.txt` file or the `logs/config-audit.log` file, you will see that a configuration element hierarchy has been created: locations, health checks, external servers, load-balancing algorithms, request processors, and subtree views.

Configuring an Entry Balancing Directory Proxy Server Deployment

An entry balancing deployment requires that the proxy server be aware of the base DN balancing point and the number of datasets in existence. An entry balancing configuration will typically involve at least two subtree views, or base DNs, defined while running the `create-initial-proxy-config` tool. One or more base DNs represent global data that is replicated across the entire set of external servers, while another base DN represents the balancing point where data is divided among replication-sets defined on the external servers.

We recommend that the necessary entry-balancing replication configuration be completed on the external servers before running the proxy `create-initial-proxy-config` tool. This means that the base DNs representing global and entry balanced data be enabled for replication and that each directory server have a replication-set-name defined. All data does not need to be populated in the directory servers at the time this tool is executed. The number of datasets may also change as time goes by. The `dsconfig` tool can be used to later add or remove configuration elements related to datasets within the proxy server configuration.

To Configure an Entry Balancing Directory Proxy Server

In this example, we assume two datasets exist among the external servers and the directory server ldap-set1-east-01 contains data for set1.

Because you are configuring an entry balancing deployment of directory server, two base DNs will be defined, the global domain of `dc=example,dc=com`, and the entry balancing domain of `ou=people,dc=example,dc=com`, which is also an entry balancing point.

1. Follow steps 1-4 in the section, *To Configure a Standard Directory Proxy Server Deployment*.

2. Enter the DN of the global domain, in this example the default value of `dc=example,dc=com`. Then, press **Enter** to accept the default of not using entry balancing on this domain.

```
Enter a DN or choose a menu item [dc=example,dc=com]:

Are entries within '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]:
```

3. Next, specify the base DN of the entry balancing domain below which the data is balanced. In this example, the entry balancing domain is `ou=people,dc=example,dc=com`. Enter `yes` to specify that data will be entry balanced.

```
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
```

4. We specify the number of backend server sets used for the entry balancing domain. In this example, we use two sets.

```
Enter a number greater than one or choose a menu item: 2
```

5. We specify that the entry balancing base DN is the `ou=people,dc=example,dc=com` DN we configured earlier.

```
Enter the entry balancing base DN or choose a menu item
 [ou=people,dc=example,dc=com]:
```

6. Define index attributes for frequently queried attributes that can be used by the global in-memory index. In this example, we add the `telephoneNumber` attribute to the global index. We also indicate that we want the attribute to be loaded into memory before the directory proxy server starts accepting connections.

```
Would you like to add attributes to the global index? (yes / no) [no]:

Enter an attribute name or choose a menu item [Press ENTER when finished entering
 index attributes]: telephoneNumber

Should the index for attribute ‘telephoneNumber’ be primed such that it is
loaded into memory before the Directory Proxy Server begins accepting connections?


  1) Yes
  2) No
```

```
  3) Yes, and all subsequent attributes
  4) No, and all subsequent attributes

Enter choice [1]: 1
```

7. Indicate whether you want to prime the RDN index at startup. The RDN index is an in-memory index that allows the directory proxy server to remember where entries exist among the datasets. It is referenced when handling modify or search requests with a base scope. Though index priming increases startup time and the load on the backend servers, it leads to better initial proxy performance.

```
Would you like to enable RDN index priming for ‘ou=people,dc=example,dc=com’? (yes /
 no) [yes]:
```

8. Enter a base DN of a new subtree view, if required.

```
Enter a DN or choose a menu item [Press ENTER when finished entering base DNs]:
```

9. Define a location for your server. You can specify multiple locations. The `create-inital-proxy-config` tool requires at least one server per location, so only specify locations for which external servers exist. Additional locations can be added afterwards using `dsconfig`. This example illustrates defining a location named east.

```
Enter a location name or choose a menu item: east
```

10. Specify the location that contains the directory proxy server itself.

```
Choose the location for this Directory Proxy Server

   1) east

   b) back
   q) quit

Enter choice [1]: 1
```

11. Define the host, port, and authentication information used by the LDAP external servers.

```
Enter a host:port or choose a menu item [localhost:389]: ldap-set1-
east-01.example.com:389
```

12. Select one or more sets of data for which this server will handle requests. Typically, each external directory server will contain data from two domains: the top-level, global domain and the entry balancing point domain, which is restricted to a specific data set.

```
Assign server ldap-set1-east-01.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
```

13. Select 3 to indicate that you want the tool to create a proxy user account on all of your LDAP external servers.

```
Would you like to prepare ldap-set1-east-01.example.com:389 for access by the Proxy
 Server?

    1)  Yes
```

```
    2)  No
    3)  Yes, and all subsequent servers
    4)  No, and all subsequent servers

Enter choice [3]:
```

14. Select the data set with which ldap-set1-east-01.example.com:389 replicates data. In this example, we select set 1.

```
  1)  ou=people,dc=example,dc=com; Server Set 1
  2)  None, data will not be replicated

Enter choice: 1
```

15. Define servers for your other server sets. Press `Enter` when you have finished.

16. Press **Enter** to complete configuring the location.

17. Review the configuration summary. Once you have confirmed that the changes are correct, press **Enter** to write the configuration.

```
  b) back
  q) quit
  w) write configuration

Enter choice [w]:
```

18. Press **Enter** to apply the changes to the directory proxy server. Alternatively, you can quit and instead run the `dsconfig` batch file at a later time. Once the changes have been applied, you cannot use the `create-initial-proxy-config` tool to configure this directory proxy server again. Instead, use the `dsconfig` tool.

```
Apply these configuration changes to the local Directory Proxy Server? (yes /no)
 [yes]:
```

Configuring the Server Using dsconfig

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

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

Using dsconfig in Interactive Command-Line Mode

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

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

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

Using dsconfig Interactive Mode: Viewing Object Menus

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

The following object menus are available:

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

To Change the dsconfig Object Menu

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

```
>>>> UnboundID® Directory Proxy Server configuration console main menu

What do you want to configure?
```

```
    1)   Account Status Notification Handler  15)  Log Retention Policy
    2)   Alert Handler                        16)  Log Rotation Policy
    3)   Backend                              17)  Password Generator
    4)   Certificate Mapper                   18)  Password Policy
    5)   Client Connection Policy             19)  Password Validator
    6)   Connection Criteria                  20)  Plugin
    7)   Connection Handler                   21)  Request Criteria
    8)   Global Configuration                 22)  Result Criteria
    9)   Identity Mapper                      23)  Root DN User
    10)  Key Manager Provider                 24)  Search Entry Criteria
    11)  Local DB Index                       25)  Search Reference Criteria
    12)  Location                             26)  Trust Manager Provider
    13)  Log Field Mapping                    27)  Virtual Attribute
    14)  Log Publisher                        28)  Work Queue

    o)   'Standard' objects are shown - change this
    q)   quit

  Enter choice:
```

Using dsconfig in Non-Interactive Mode

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

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

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

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

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

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

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

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

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

To Get the Equivalent dsconfig Non-Interactive Mode Command

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

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

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

Using dsconfig Batch Mode

The UnboundID® Directory Proxy Server provides a `dsconfig` batching mechanism that reads multiple `dsconfig` invocations from a file and executes them sequentially. The batch file provides advantages over standard scripting by minimizing LDAP connections and JVM invocations that normally occur with each `dsconfig` call. Batch mode is the best method to use with setup scripts when moving from a development environment to test environment, or from a test environment to a production environment.

If a `dsconfig` command has a missing or incorrect argument, the command will fail and abort the batch process. Any command that was successfully executed prior to the abort will be applied to the Directory Proxy Server. The `--no-prompt` option is required with `dsconfig` in batch mode.

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

The Directory Proxy Server also provides a `docs/sun-ds-compatibility.dsconfig` file for migrations from Sun/Oracle Directory Server Enterprise Edition (DSEE) to UnboundID® Directory Proxy Server machines.

To Configure the Server in dsconfig Batch Mode

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

```
dsconfig create-global-attribute-index
--processor-name ou_people_dc_example_dc_com-eb-req-processor
--index-name exAccountNumber --set prime-index:true
--set guaranteed-unique:true

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

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

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

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

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

```
root@proxy-01:bin/dsconfig --no-prompt \
  --bindDN "cn=directory manager" --bindPassword password \
  --port 389 --batch-file ../dsconfig.post-setup
```

```
""


Batch file '../dsconfig.post-setup' contains 3 commands
Executing: create-global-attribute-index --no-prompt --bindDN "cn=directory manager"
 --bindPassword ******** --port 1389
--processor-name ou_people_dc_example_dc_com-eb-req-processor --index-name
 exAccountNumber --set prime-index:true
--set guaranteed-unique:true
Executing: create-placement-algorithm --no-prompt
--bindDN "cn=directory manager" --bindPassword ******** --port 1389
--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"
Executing: delete-placement-algorithm --no-prompt
--bindDN "cn=directory manager" --bindPassword ******** --port 1389
--processor-name ou_people_dc_example_dc_com-eb-req-processor --algorithm-name round-
robin
```

```
""""""
```

Generating a Summary of Configuration Components

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

By default, the tool generates a list of basic components. To include a list of advanced components, use the `--advanced` option. To run the tool on an offline server, use the `--offline` option. Run the `summarize-config --help` option to view other available tool options.

To Generate a Summary of Configuration Components

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

```
$ bin/summarize-config
```

```
Global Configuration:
  location:
    Location: Houston
      preferred-failover-location: none

Client Connection Policies:
  Client Connection Policy: default
     subtree-view:
      Subtree View: dc_example_dc_com-view
        base-dn: "dc=example,dc=com"
        request-processor:
           Entry Balancing Request Processor: dc_example_dc_com-eb-req-processor
             enabled: true
             allowed-operation: abandon, add, bind, compare, delete, extended,
             modify, modify-dn, search
             entry-balancing-base-dn: "dc=example,dc=com"
             global-index-memory-percent: 70
             log-index-duplicates: false
             duplicate-entry-search: all-servers
             broadcast-updates-above-balancing-point: false
             subordinate-request-processor:
               Proxying Request Processor: dc_example_dc_comServer_Set_1-req-
                 processor
                 enabled: true
                 allowed-operation: abandon, add, bind, compare, delete, extended,
                 modify, modify-dn, search
                 referral-behavior: pass-through
                 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,
                 operation-purpose, 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
                 supported-control-oid: -
                 load-balancing-algorithm:
                   Fewest Operations Load Balancing Algorithm:
                   dc_example_dc_comServer_Set_1-fewest-operations
                     enabled: true
                     use-location: true
                     prefer-degraded-servers-over-failover: false
                     maximum-allowed-local-response-time: 30 s
```

```
                    maximum-allowed-nonlocal-response-time: 30 s
                    maximum-retry-count: 1
                    server-affinity-provider: none
                    backend-server:
                      UnboundID DS External Server: localhost:2389
                        server-host-name: localhost
                        server-port: 2389
                        bind-dn: "cn=Proxy User,cn=Root DNs,cn=config"
                        password: ********
                        connection-security: none
                        authentication-method: simple
                        health-check-frequency: 30 s
                     allowed-operation: abandon, add, bind, compare, delete,
                     extemded, modify, modify-dn, search
                        load-schema: true
                        health-check:
                          Search LDAP Health Check:
                          localhost:2389_dc_example_dc_com-search-health-check
                            enabled: true
                            use-for-all-servers: false
                            base-dn: "dc=example,dc=com"
                            scope: base-object
                            filter: (objectClass=*)
                            maximum-local-available-response-time: 1 s
                            maximum-nonlocal-available-response-time: 1 s
                            minimum-local-degraded-response-time: 500 ms
                            minimum-nonlocal-degraded-response-time: 500 ms
                            maximum-local-degraded-response-time: 10 s
                            maximum-nonlocal-degraded-response-time: 10 s
                            minimum-local-unavailable-response-time: 5 s
                            minimum-nonlocal-unavailable-response-time: 5 s
                            allow-no-entries-returned: true
                            allow-multiple-entries-returned: true
                            available-filter: -
                            degraded-filter: -
                            unavailable-filter: -
                        location:
                          Location: Houston
                            preferred-failover-location: none
                        trust-manager-provider: none
                        key-manager-provider: none
                  transformation: none
                Proxying Request Processor: dc_example_dc_comServer_Set_2-req-
                  processor
                  enabled: true
                  allowed-operation: abandon, add, bind, compare, delete,
                  extended, modify, modify-dn, search
                  referral-behavior: pass-through
                  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,
                  operation-purpose, 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
                  supported-control-oid: -
                  load-balancing-algorithm:
                    Fewest Operations Load Balancing Algorithm:
                    dc_example_dc_comServer_Set_2-fewest-operations
                      enabled: true
                      use-location: true
                      prefer-degraded-servers-over-failover: false
                      maximum-allowed-local-response-time: 30 s
                      maximum-allowed-nonlocal-response-time: 30 s
                      maximum-retry-count: 1
                      server-affinity-provider: none
                      backend-server:
                        UnboundID DS External Server: localhost:3389
                          server-host-name: localhost
                          server-port: 3389
                          bind-dn: "cn=Proxy User,cn=Root DNs,cn=config"
                          password: ********
                          connection-security: none
                          authentication-method: simple
                          health-check-frequency: 30 s
                        allowed-operation: abandon, add, bind, compare, delete,
                        extended, modify, modify-dn, search
                           load-schema: true
                           health-check:
```

```
                        Search LDAP Health Check:
                        localhost:3389_dc_example_dc_com-search-health-check
                          enabled: true
                          use-for-all-servers: false
                          base-dn: "dc=example,dc=com"
                          scope: base-object
                          filter: (objectClass=*)
                          maximum-local-available-response-time: 1 s
                          maximum-nonlocal-available-response-time: 1 s
                          minimum-local-degraded-response-time: 500 ms
                          minimum-nonlocal-degraded-response-time: 500 ms
                          maximum-local-degraded-response-time: 10 s
                          maximum-nonlocal-degraded-response-time: 10 s
                          minimum-local-unavailable-response-time: 5 s
                          minimum-nonlocal-unavailable-response-time: 5 s
                          allow-no-entries-returned: true
                          allow-multiple-entries-returned: true
                          available-filter: -
                          degraded-filter: -
                          unavailable-filter: -
                      location:
                        Location: Houston
                          preferred-failover-location: none
                      trust-manager-provider: none
                      key-manager-provider: none
          transformation: none
    Global Attribute Indexes:
      Global Attribute Index: uid
        attribute: uid
        prime-index: true
        guaranteed-unique: true
    Placement Algorithms:
      Round Robin Placement Algorithm: round-robin
        enabled: true
```

Configuring Server Groups

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

About the Server Group Example

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

The following example procedure adds three directory server instances into the server group labelled "group-one". The commands are run on server1.example.com.

Server	Host Name	LDAP Port
instance 1	server1.example.com	1389

Server	Host Name	LDAP Port
instance 2	server2.example.com	2389
instance 3	server3.example.com	3389

To Create a Server Group

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

```
$ bin/dsframework create-group --groupName group-one \
  --description "Server Group One"
```

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

```
$ bin/dsframework register-server \
  --serverID server1.example.com:1389 \
  --set hostname:server1.example.com \
  --set ldapport:1389 --set ldapEnabled:true

$ bin/dsframework register-server \
  --serverID server2.example.com:2389 \
  --set hostname:server2.example.com \
  --set ldapport:2389 --set ldapEnabled:true

$ bin/dsframework register-server \
  --serverID server3.example.com:3389 \
  --set hostname:server3.example.com \
  --set ldapport:3389 --set ldapEnabled:true
```

3. Add each directory server to the group.

```
$ bin/dsframework add-to-group \
  --groupName group-one \
  --memberName server1.example.com:1389

$ bin/dsframework add-to-group \
  --groupName group-one \
  --memberName server2.example.com:2389

$ bin/dsframework add-to-group \
  --groupName group-one \
  --memberName server3.example.com:3389
```

4. Set a global configuration property using the `dsconfig` tool.

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

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

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

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

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

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

7. Repeat the step 6 on the second and third directory server instance.

8. Test the server group by disabling the log publisher on the first directory server instance by using the `--applyChangeTo single-server`.

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

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

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

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

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

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

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

Managing Root Users Accounts

The UnboundID® Directory Proxy Server provides a default root user, `cn=Directory Manager`, that is stored in the server's configuration file (for example, under `cn=Root DNs,cn=config`). The root user is the LDAP-equivalent of a UNIX super-user account and inherits its read-write privileges from the default root privilege set. Root user entries are stored in the server's configuration and not in backend data. Root users have access to all of the data in the directory.

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

Default Root Privileges

The UnboundID® Directory Proxy Server contains a privilege subsystem that allows for a more fine-grained control of privilege assignments. The following set of root privileges are available to each root user DN:

Table 1: 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.
modify-acl	Allows the user to modify access control rules.
password-reset	Allows the user to reset user passwords but not their own. The user must also have privileges granted by access control to write the user password to the target entry.
privilege-change	Allows the user to change the set of privileges for a specific user, or to change the set of privileges automatically assigned to a root user.
server-restart	Allows the user to request a server restart.
server-shutdown	Allows the user to request a server shutdown.
soft-delete-read	Allows the user access to soft-deleted entries.
stream-values	Allows the user to perform a stream values extended operation that obtains all entry DNs and/or all values for one or more attributes for a specified portion of the DIT.
third-party-task	Allows the associated user to invoke tasks created by third-party developers.
unindexed-search	Allows the user to perform an unindexed search in the Oracle Berkeley DB Java Edition backend.
update-schema	Allows the user to update the server schema.
use-admin-session	Allows the associated user to use an administrative session to request that operations be processed using a dedicated pool of worker threads.

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

Table 2: Other Available Privileges

Privilege	Description
bypass-read-aci	Allows the associated user to bypass access control checks performed by the server for bind, compare, and search operations. Access control evaluation may still be enforced for other types of operations.
jmx-notify	Allows the associated user to subscribe to receive JMX notifications.
jmx-read	Allows the associated user to perform JMX read operations.
jmx-write	Allows the associated user to perform JMX write operations.

To View the Default Root User Privileges Using dsconfig

The root DN accounts are the only user accounts that are stored within the Directory Proxy Server's configuration under `cn=Root DNs,cn=config`. You can view the default privileges automatically granted to root users using the `dsconfig` tool.

- Use `dsconfig` to view the root DN.

```
$ bin/dsconfig get-root-dn-prop
```

To Modify the Root User Password

Root users are governed by the Root Password Policy and by default, their passwords never expire. However, in the event that you want to change a root user's password, you can use the `ldappasswordmodify` tool.

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

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

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

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

3. Remove the text file.

```
$ rm rootuser.txt
```

To Create a Root User

You can create another root user by adding an entry under `cn=Root DNs,cn=config`. By default, the new root user will receive the default set of root privileges. If you want the new root user to have a limited set of privileges, you can remove the privileges using the `dsconfig` tool.

1. Open a text editor, and create a file containing the root user entry.

```
dn: cn=Directory Manager2,cn=Root DNs,cn=config
objectClass: top
objectClass: person
objectClass: organizationalPerson
objectClass: inetOrgPerson
```

```
objectClass: ds-cfg-root-dn-user
userPassword: password
cn: Directory Manager2
sn: Manager2
ds-cfg-alternate-bind-dn: cn=Directory Manager2
givenName: Directory
```

2. Use `ldapmodify` to add the entry.

```
$ bin/ldapmodify --port 1389 --bindDN "cn=Directory Manager" \
  --bindPassword secret --defaultAdd --filename "rootuser.ldif"
```

Modifying Individual Root User Privileges

All root users automatically inherit the default root privileges defined in the `default-root-privilege-name` configuration property. However, you can give individual root users additional privileges that are not included in the set of default root privileges. You can also remove default root privileges from individual root users.

Modifying the privileges of the root user can be accomplished by adding the `ds-privilege-name` operational attribute to the entry for the root user. Any values containing a privilege name will grant that privilege to the user in addition to the set of default root privileges. Any values containing a minus sign followed by a privilege name will remove that privilege from that root user, even if it is included in the set of default root privileges.

1. Open a text editor, and create a file containing the changes to the root user entry. The following example grants the `proxied-auth` privilege and removes the `server-shutdown` and `server-restart` privileges.

```
dn: cn=Directory Manager2,cn=Root DNs,cn=config
changetype: modify
add: ds-privilege-name
ds-privilege-name: proxied-auth
ds-privilege-name: -server-shutdown
ds-privilege-name: -server-restart
```

2. Use `ldapmodify` to apply the change.

```
$ bin/ldapmodify --port 1389 --bindDN "cn=Directory Manager" \
  --bindPassword secret --filename "modifyRootUserPrivileges.ldif"
```

Configuring Locations

UnboundID® Directory Proxy Server defines locations, both for the LDAP external servers and the proxy server instances themselves. A location defines a collection of servers that share access and latency characteristics. For example, your deployment might include two data centers, one in the east and one in the west. These data centers would be configured as two locations in the 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 Directory Proxy Management Console or the command line.

The Directory Proxy Server itself is also associated with a location. This location is specified in the global configuration properties of the directory proxy server. If the load balancing

algorithm's `use-location` property is set to true, then the load balancing component of the directory proxy server refers to the directory proxy server's location to determine the external servers it prefers to communicate with.

To Configure Locations Using dsconfig

1. Use the `dsconfig` tool to configure the LDAP external server locations.

```
$ bin/dsconfig
```

2. Type the hostname or IP address for your directory proxy server, or press **Enter** to accept the default, localhost.

```
Directory Proxy Server hostname or IP address [localhost]:
```

3. Type the number corresponding how you want to connect to the directory proxy server, or press **Enter** to accept the default, LDAP.

```
How do you want to connect?
  1) LDAP
  2) LDAP with SSL
  3) LDAP with StartTLS
```

4. Type the port number for your directory proxy server, or press **Enter** to accept the default, 389.

```
Directory Proxy Server port number [389]:
```

5. Type the administrator's bind DN or press **Enter** to accept the default (cn=Directory Manager), and then type the password.

```
Administrator user bind DN [cn=Directory Manager]:
Password for user 'cn=Directory Manager':
```

6. In the Directory Proxy Server configuration console menu, enter the number corresponding to location configuration. Then, enter the number to create a new location.

```
Enter choice: 8

>>>> Location management 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]: 2
```

7. Enter the name of the new location. This example demonstrates configuring a location called East. Enter **f** to finish configuring the location. Repeat this procedure to create a location called West.

```
>>>>  Enter a name for the location that you want to create: east
```

```
>>>>  Configure the properties of the location

      Property                   Value(s)
      ---------------------------------
   1) description                 -
   2) preferred-failover-location  -

   ?) help
   f) finish – create the new location
   d) display the equivalent dsconfig arguments to
      create this object
   b) back
   q) quit

Enter choice [b]: f
```

8. Next, edit the configuration of an existing location, in this example a location named East.

```
>>>> Location management menu
What would you like to do?

  1) List existing locations
  2) Create a new location
  3) View and edit an existing location
  4) Delete an existing location

  b) back
  q) quit

Enter choice [b]: 3

>>>> Select the location from the following list:
   1) East
   2) West

   b) back
   q) quit

Enter choice [b]: 1
```

9. Define the preferred failover location property for East. This property provides alternate locations that can be used if servers in this location are not available. If more than one location is provided, the directory proxy server tries the locations in the order listed.

```
 >>>> Configure the properties of the Location

     Property                   Value(s)
     ---------------------------------
  1) description                 -
  2) preferred-failover-location  -

  ?) help
  f) finish – create the new location
  d) display the equivalent dsconfig arguments to create this object
  b) back
  q) quit

Enter choice [b]: 2

...

Do you want to modify the 'preferred-failover-location' property?

   1) Add one or more values

   ?) help
   q) quit

Enter choice [1]: 2

Select the locations you wish to add:

  1) East
  2) West
```

```
  3) Create a new location
  4) Add all locations

...

Enter one or more choices separated by commas[b]: 2
```

10. Verify and apply your change to the property.

```
Do you want to modify the 'preferred-failover-location' property?

  1) Use the value: West
  2) Add one or more values
  3) Remove one or more values
  4) Leave undefined
  5) Revert changes

  ?) help
  q) quit

Enter choice [1]:

>>>> Configure the properties of the location

     Property                    Value(s)
     ----------------------------------
  1) description                  -
  2) preferred-failover-location  West

  ?) help
  f) finish – apply any changes to the Location
  d) display the equivalent dsconfig command lines to either
     re-create this object or only to apply pending changes
  b) back
  q) quit

Enter choice [b]: f
```

11. Repeat steps 8 and 9 for the West location, assigning it a failover location of East.

To Modify Locations Using dsconfig

1. Use the `dsconfig` tool to configure the LDAP external server locations.

```
$ bin/dsconfig
```

2. Type the hostname or IP address for your directory proxy server, or press **Enter** to accept the default, localhost.

```
Directory Proxy Server hostname or IP address [localhost]:
```

3. Type the number corresponding how you want to connect to the directory proxy server, or press **Enter** to accept the default, LDAP.

```
How do you want to connect?
  1) LDAP
  2) LDAP with SSL
  3) LDAP with StartTLS
```

4. Type the port number for your directory proxy server, or press **Enter** to accept the default, 389.

```
Directory Proxy Server port number [389]:
```

5. Type the administrator's bind DN or press **Enter** to accept the default (cn=Directory Manager), and then type the password.

```
Administrator user bind DN [cn=Directory Manager]:
Password for user 'cn=Directory Manager':
```

6. In the Directory Proxy Server configuration console menu, enter the number corresponding to global configuration. Then enter the number to view and edit the global configuration.

```
>>>> Global configuration management menu
What would you like to do?

  1) View and edit the Global Configuration

  b) back
  q) quit

Enter choice [b]: 1
```

7. Enter the number associated with the location configuration property.

```
Enter choice [b]: 2
```

8. Specify a new location for this proxy server instance, in this example the East location. Operations involving communications with other servers may prefer servers in the same location to ensure low-latency responses.

```
>>>> Configuring the 'location' property

...

Do you want to modify the 'location' property?

  1) Leave undefined
  2) Change it to the location: East
  3) Change it to the location: West
  4) Create a new location

  b) back
  q) quit

Enter choice [b]: 2
```

9. Enter **f** to finish the operation.

```
Enter choice [b]: f
```

Configuring Server Health Checks

You can use the UnboundID® Directory Proxy Server to configure different types of health checks for your deployment. The health checks define external server availability as either being available, unavailable, or degraded. The external server health is given a value from 0 to 10, which is used to determine if the server is available and how that server compares to other servers with the same state. Load-balancing algorithms can be used to check the score and prefer servers with higher scores over those with lower scores.

An individual health check can be defined for use against all external servers or assigned to individual external servers, as determined by the `use-for-all-servers` parameter within

the health check configuration object. If `use-for-all-servers` is set to true, the directory proxy server applies the health check to all external servers in all locations. If `use-for-all-servers` is set to false, then the health check is only employed against an external server if the configuration object for that external server lists the health check.

For more information about health checks and the type of health checks supported by UnboundID® Directory Proxy Server, see *About LDAP Health Checks*.

About the Default Health Checks

By default, the directory proxy server has two health check instances enabled for use on all servers:

- **Consume Admin Alerts**. This health check detects administrative alerts as soon as they are issued and then checks to see if the alert is associated with a server becoming degraded or unavailable. The directory proxy server accomplishes this by using an LDAP persistent search against the directory server.
- **Get Root DSE**. This health check detects if the root DSE entry exists on the LDAP external server. As this entry always exists on an UnboundID® Directory Server, the absence of the entry suggests that the LDAP external server may be degraded or unavailable.

About Creating a Custom Health Check

You can create a new health check from scratch or use an existing health check as a template for the configuration of a new health check. If you choose to create a custom health check, you can create one of the following types:

- **Admin Alert Health Check**. This health check watches for administrative alerts generated by the LDAP external server to determine whether the server has entered a degraded or unavailable state.
- **Groovy Scripted LDAP Health Check**. This health check allows you to create custom LDAP health checks in a dynamically-loaded Groovy script, which implements the `ScriptedLDAPHealthCheck` class defined in the Server SDK.
- **Replication Backlog Health Check**. While the Admin Alert Health Check consumes replication backlog alerts emitted from external servers, a finer definition of external server health based on replication backlog can be defined with this health check. If a server falls too far behind in replication, then the directory proxy server can stop sending requests to it. A server is classified as degraded or unavailable if the threshold is reached for the number of backlogged changes, the age of the oldest backlogged change, or both.
- **Search LDAP Health Check**. This health check performs searches on an LDAP external server and gauges the health of the server depending if the expected results were returned within an acceptable response time. For example, if an error occurs while attempting to communicate with the server, then the server is considered unavailable. You can also apply filters to the results to use values within the monitor entry as indicators of server health.

- **Third Party LDAP Health Check**. This health check allows you to define LDAP health check implementations in third-party code using the Server SDK.
- **Work Queue Busyness Health Check**. This health check may be used to monitor the percentage of time that worker threads in backend servers spend processing requests.

To Configure a Health Check Using dsconfig

1. Use the `dsconfig` tool to configure the LDAP external server locations.

```
$ bin/dsconfig
```

2. Type the hostname or IP address for your directory proxy server, or press **Enter** to accept the default, localhost.

```
Directory Proxy Server hostname or IP address [localhost]:
```

3. Type the number corresponding how you want to connect to the directory proxy server, or press **Enter** to accept the default, LDAP.

```
How do you want to connect?
  1) LDAP
  2) LDAP with SSL
  3) LDAP with StartTLS
```

4. Type the port number for your directory proxy server, or press **Enter** to accept the default, 389.

```
Directory Proxy Server port number [389]:
```

5. Type the administrator's bind DN or press **Enter** to accept the default (cn=Directory Manager), and then type the password.

```
Administrator user bind DN [cn=Directory Manager]:
Password for user 'cn=Directory Manager':
```

6. In the Directory Proxy Server configuration console menu, enter the number corresponding to LDAP health checks. Enter the number to create a new LDAP Health Check, then press n to create a new health check from scratch.

```
>>>> LDAP Health Check management menu

What would you like to do?

  1) List existing LDAP Health Checks
  2) Create a new LDAP Health Check
  3) View and edit an existing LDAP Health Check
  4) Delete an existing LDAP Health Check

  b) back
  q) quit

Enter choice [b]: 2

>>>> Select an existing LDAP Health Check to use as a template for the new
LDAP Health Check configuration or ‘n’ to create one from scratch:

  1) Consume Admin Alters
  2) Get Root DSE

  n) new LDAP Health Check created from scratch
```

```
  c) cancel
  q) quit

 Enter choice [n]: n
```

7. Select the type of health check you want to create. This example demonstrates the creation of a new search LDAP health check.

```
>>> Select the type of LDAP Health Check that you want to create:

     1)  Admin Alert LDAP Health Check
     2)  Custom LDAP Health Check
     3)  Groovy Scripted LDAP Health Check
     4)  Replication Backlog LDAP Health Check
     5)  Search LDAP Health Check
     6)  Third Party LDAP Health Check
     7)  Work Queue Busyness LDAP Health Check

     ?) help
     c) cancel
     q) quit

Enter choice [c]: 5
```

8. Specify a name for the new health check. In this example, the health check is named `Get example.com`.

```
>>>> Enter a name for the search LDAP Health Check that you want to create: Get
 example.com
```

9. Enable the new health check.

```
>>>> Configuring the 'enabled' property

Indicates whether this LDAP health check is enabled for use in the server.

Select a value for the 'enabled' property:

   1) true
   2) false

   ?) help
   c) cancel
   q) quit

Enter choice [c]: 1
```

10. Next, configure the properties of the health check. You may need to modify the `base-dn` property, as well as one or more response time thresholds for non-local external servers, accommodating WAN latency. Below is a Search LDAP Health Check for the single entry `dc=example,dc=com`, which allows non-local responses of up to 2 seconds to still be considered healthy.

```
>>>> Configure the properties of the Search LDAP Health Check

           Property                                  Value(s)
           ----------------------------------------------------------
     1)    description                               -
     2)    enabled                                   true
     3)    use-for-all-servers                       false
     4)    base-dn                                   "dc=example,dc=com"
     5)    scope                                     base-object
     6)    filter                                    (objectClass=*)
     7)    maximum-local-available-response-time     1 s
     8)    maximum-nonlocal-available-response-time  2 s
     9)    minimum-local-degraded-response-time      500 ms
     10)   minimum-nonlocal-degraded-response-time   1 s
     11)   maximum-local-degraded-response-time      10 s
     12)   maximum-nonlocal-degraded-response-time   10 s
```

```
     13)  minimum-local-unavailable-response-time    5 s
     14)  minimum-nonlocal-unavailable-response-time 5 s
     15)  allow-no-entries-returned                  true
     16)  allow-multiple-entries-returned            true
     17)  available-filter                           -
     18)  degraded-filter                            -
     19)  unavailable-filter                         -

     ?) help
     f) finish - create the new Search LDAP Health Check
     d) display the equivalent dsconfig arguments to create this object
     b) back
     q) quit
```

Configuring LDAP External Servers

The LDAP external server configuration element defines the connection, location, and health check information necessary for the directory proxy server to communicate with the server properly.

UnboundID® Directory Proxy Server includes a tool, `prepare-external-server`, for configuring communication between the directory proxy server and the LDAP backend server. After you add a new LDAP external server to an existing installation, we strongly recommend that you run this script to automatically create the user account necessary for communications. The `prepare-external-server` tool does not make configuration changes to the local directory proxy server, only the external server is modified. When you run this tool, you must supply the user account and password that you specified for the directory proxy server during configuration, `cn=Proxy User` by default.

Important: You should not use `cn=Directory Manager` as the account to use for communication between the directory proxy server and the directory server. For security reasons, the account used to communicate between the directory proxy server and the directory server should not be directly accessible by clients accessing the directory proxy server. The account that you choose should meet the following criteria:

- For all server types, it should not exist in the directory proxy server but only in the backend directory server instances.
- For UnboundID Directory Server and Alcatel-Lucent 8661 Directory Server, this user should be a root user.
- For UnboundID Directory Server and Alcatel-Lucent 8661 Directory Server, this user should not automatically inherit the default set of root privileges, but instead should have exactly the following set of privileges: `bypass-read-acl`, `config-read`, `lockdown-mode`, `proxied-auth`, and stream-values.
- For Sun Directory Servers, the account should be created below the `cn=Root DNs,cn=config` entry and the `nsSizeLimit`, `nsTimeLimit`, `nsLookThroughLimit`, and `nsIdleTimeout` values for the account should be set to -1. You also need to create access control rules to grant the

user account appropriate permissions within the server. The `prepare-external-server` tool handles all of this work automatically.

To Configure an External Server Using dsconfig

1. Use the `dsconfig` tool to create and configure external servers. Then, specify the hostname, connection method, port number, and bind DN as described in previous procedures.

```
$ bin/dsconfig
```

2. In the Directory Proxy Server configuration console menu, enter the number corresponding to external servers. Then, enter the number to create a new external server.

```
>>>> External server management menu
What would you like to do?

  1) List existing external servers
  2) Create a new external server
  3) View and edit an existing external server
  4) Delete an existing external server

  b) back
  q) quit

Enter choice [b]: 2
```

3. Select the type of server you want to create. This example creates a new UnboundID Directory Server.

```
>>>> Select the type of external server that you want to create:

  1) UnboundID DS external server
  2) JDBC external server
  3) LDAP external server
  4) Sun DS external server

  ?) help
  c) cancel
  q) quit

Enter choice [c]: 1
```

4. Specify a name for the new external server. In this example, the external server is named east1.

```
>>>> Enter a name for the UnboundID DS external server that you want
to create: east1
```

5. Configure the host name or IP address of the target LDAP external server.

```
Enter a value for the 'server-host-name' property:east1.example.com
```

6. Next, configure the location property of the new external server.

```
Do you want to modify the 'location' property?

   1) Leave undefined
   2) Change it to the location: East
   3) Change it to the location: West
   4) Create a new location
```

```
   ?) help
   q) quit

Enter choice [1]: 2
```

7. Next, define the bind DN and bind password.

```
Do you want to modify the 'bind-dn' property?

   1) Leave undefined
   2) Change the value

   ?) help
   q) quit

Enter choice [1]: 2

Enter a value for the 'bind-dn' property [continue]: cn=Proxy User,cn=Root
 DNs,cn=config

Enter choice [b]: 6

...

Do you want to modify the 'password' property?

   1) Leave undefined
   2) Change the value
   ?) help
   q) quit

Enter choice [1]: 2

Enter a value for the 'password' property [continue]:
Confirm the value for the 'password' property:
```

8. Enter `f` to finish the operation.

```
Enter choice [b]: f

The UnboundID DS external server was created successfully.
```

   Once you have completed adding the server, run the `prepare-external-server` tool to configure communications between the directory proxy server and the UnboundID Directory Servers.

About the prepare-external-server Tool

Use the `prepare-external-server` tool if you have added LDAP external servers using dsconfig. The `create-initial-proxy-config` tool automatically runs the `prepare-external-server` tool to configure server communications so that you do not need to invoke it separately. The `create-initial-proxy-config` tool verifies that the proxy user account exists and has the correct password and required privileges. If it detects any problems, it prompts for manager credentials to rectify them.

If you want the `prepare-external-server` tool to add the LDAP external server's certificates to the Directory Proxy Server's trust store, you must include the `--proxyTrustStorePath` option, and either the `--proxyTrustStorePassword` or the `--proxyTrustStorePasswordFile` option. The default location of the directory proxy server trust store is `config/truststore`. The pin is encoded in the `config/truststore.pin` file.

For example, run the tool as follows to prepare an UnboundID® Directory Server on the remote host, ds-east-01.example.com, listening on port 1389 for access by the Directory Proxy Server using the default user account cn=Proxy User:

```
prepare-external-server --hostname ds-east-01.example.com \
--port 1389 --baseDN dc=example,dc=com --proxyBindPassword secret
```

When the `prepare-external-server` command above is executed, it creates the `cn=Proxy User` Root DN entry as well as an access control rule in the Directory Server to grant the proxy user the proxy access right.

To Configure Server Communication Using the prepare-external-server Tool

The following example illustrates how to run the `prepare-external-server` tool to prepare a Directory Server on the remote host, ds-east-01.example.com, listening on port 1636. The directory server is being accessed by a directory proxy server that uses the default user account `cn=Proxy User,cn=Root DNs,cn=config`. Since a password to the truststore is not provided, the truststore defined in the `--proxyTrustStorePath` is referenced in a read-only manner.

- Use the `prepare-external-server` tool to prepare a directory server. Follow the prompts to set up the external server.

```
$ ./UnboundID-Proxy/bin/prepare-external-server \
  --baseDN dc=example,dc=com
  --proxyBindPassword password \
  --hostname ds-east-01.example.com \
  --useSSL \
  --port 1636
  --proxyTrustStorePath /full/path/to/trust/store \
  --proxyTrustStorePassword secret
```

```
Testing connection to ds-east-01.example.com:1636 .....

Do you wish to trust the following certificate?

Certificate Subject: CN=ds-east-01.example.com, O=Example Self-Signed Certificate
Issuer Subject:      CN=ds-east-01.example.com, O=Example Self-Signed Certificate
Validity:            Thu May 21 08:02:30 CDT 2009 to Wed May 16 08:02:30 CDT 2029

Enter 'y' to trust the certificate or 'n' to reject it.

y

The certificate was added to the local trust store

Done
Testing 'cn=Proxy User' access to ds-east-01.example.com:1636 ..... Failed to bind as
'cn=Proxy User'

Would you like to create or modify root user 'cn=Proxy User' so that it is available
for this 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 Load Balancing

You can distribute the load on your proxy server using one of the load-balancing algorithms provided with UnboundID® Directory Proxy Server. By default, the directory proxy server will prefer local servers over non-local servers, unless you set the `use-location` property of the load-balancing algorithm to false. Within a given location, the directory proxy server prefers available servers over degraded servers. This means that if at all possible, the directory proxy server will send requests to servers that are local and available before considering selecting any server that is non-local or degraded.

Note: If the `use-location` property is set to true, then the load is balanced only among available external servers in the same location. If no external servers are available in the same location, the directory proxy server will attempt to use available servers in the first preferred failover location, and so on. The failover based on no external servers with AVAILABLE health state can be customized to allow the directory proxy server to prefer local DEGRADED health servers to servers in a failover location. See the UnboundID® Directory Proxy Server Reference Guide for more information on the `prefer-degraded-servers-over-failover` property.

The directory proxy server provides the following load-balancing algorithms:

- **Failover load balancing**. This algorithm forwards requests to servers in a given order, optionally taking the location into account. If the preferred server is not available, then it will fail over to the alternate server in a predefined order. This balancing method can be useful if certain operations, such as LDAP writes, need to be forwarded to a primary external server, with secondary external servers defined for failover if necessary.
- **Fewest operations load balancing**. This algorithm forwards requests to the backend server with the fewest operations currently in progress and tends to exhibit the best performance.
- **Health weighted load balancing**. This algorithm assigns weights to servers based on their health scores and, optionally, their locations. For example, servers with a higher health check score will receive a higher proportion of the requests than servers with lower health check scores.
- **Round robin load balancing**. This algorithm spreads the work evenly across a set of servers and is historically the most common approach to load balancing.
- **Single server load balancing**. This algorithm forwards all operations to a single external server that you specify.
- **Weighted load balancing**. This algorithm uses statically defined weights for sets of servers to divide load among external servers. External servers are grouped into weighted sets, the values of which, when added to all of the weighted sets for the load balancing algorithm, represent a percentage of the load the external servers should receive.

- **Criteria based load balancing**. This algorithm allows you to balance your load across a server topology depending on the types of operations received or the client issuing the request.

  For example, ds1 and ds2 are assigned to a weighted set named Set-80 and assigned the weight 80. The external servers ds3 and ds4 are assigned to the weighted set Set-20 and assigned the weight 20. When both sets, Set-80 and Set-20, are assigned to the load balancing algorithm, 80 percent of the load will be forwarded to ds1 and ds2, while the remaining 20 percent will be forwarded to ds3 and ds4.

To Configure Load Balancing Using dsconfig

1. Use the `dsconfig` tool to create and configure a load-balancing algorithm.

   ```
   $ bin/dsconfig
   ```

   Specify the hostname, connection method, port number, and bind DN as described in previous procedures.

2. In the Directory Proxy Server configuration console menu, enter the number associated with load-balancing algorithms.

   ```
   >>>> Load-Balancing Algorithm management menu

   What would you like to do?

     1) List existing Load-Balancing Algorithms
     2) Create a new Load-Balancing Algorithm
     3) View and edit an existing Load-Balancing Algorithm
     4) Delete an existing Load-Balancing Algorithm

     ?) help
     q) quit

   Enter choice [1]: 2
   ```

3. Select an existing load-balancing algorithm to use as a template or select `n` to create a new load-balancing algorithm from scratch.

   ```
   >>>>Choose how to create the new Load Balancing Algorithm:

   n) new Load Balancing Algorithm created from scratch
   t) use an existing Load Balancing Algorithm as a template
   b) back
   q) quit

   Enter a choice [n]: n
   ```

4. Select the type of load-balancing algorithm that you want to create. Depending on type of algorithm you select, you will be guided through a series of configuration properties, such as providing a name and selecting an LDAP external server.

   ```
   >>>> Select the type of Load Balancing Algorithm that you want to
   create:

       1)  Failover Load Balancing Algorithm
       2)  Fewest Operations Load Balancing Algorithm
       3)  Health Weighted Load Balancing Algorithm
       4)  Round Robin Load Balancing Algorithm
       5)  Single Server Load Balancing Algorithm
       6)  Weighted Load Balancing Algorithm
   ```

```
    ?)  help
    c)  cancel
    q)  quit

Enter choice [c]: 3
```

5. Review the configuration properties for your new load-balancing algorithm. If you are satisfied, enter `f` to finish.

Configuring Criteria Based Load Balancing Algorithms

You can configure alternate load balancing algorithms that determine how they function according to request or connection criteria. These algorithms allow you to balance your load across a server topology depending on the types of operations received or the client issuing the request. They are called criteria-based load balancing algorithms and are configured using at least one connection criteria or request criteria. For example, you can configure criteria-based load balancing algorithms to accomplish the following:

- Route write operations to a single server from a set of replicated servers, to prevent replication conflicts, while load balancing all other operations across the full set of servers.
- Route all operations from a specific client to a single server in a set of replicated servers, eliminating errors that arise from replication latency, while load balancing operations from other clients across the full set of servers. This configuration is useful for certain provisioning applications that need to write and then immediately read the same data.

When a request is received, the proxying request processor first iterates through all of the criteria-based load balancing algorithms in the order in which they are listed, to determine whether the request matches the associated criteria. If there is a match, then the criteria-based load balancing algorithm is selected. If there is not a match, then the default load-balancing algorithm is used.

Preferring Failover LBA for Write Operations

An administrator can configure the Directory Proxy Server to use Criteria-Based Load Balancing Algorithms to strike a balance between providing a consistent view of directory data for applications that require it and taking advantage of all servers in a topology for handling read-only operations, such as search and bind. The flexible configuration model supports a wide range of criteria for choosing which Load Balancing Algorithm to use for each operation. In most Directory Proxy Server deployments, UnboundID recommends using a Failover Load Balancing Algorithm for at least ADD, DELETE, and MODIFY-DN operations if not for all types of write operations.

Each Proxying Request Processor configured in the Directory Proxy Server uses a Load Balancing Algorithm to choose which Directory Server to use for a particular operation. The Load Balancing Algorithm takes several factors into account when choosing a server:

- The availability of the Directory Servers.
- The location of the Directory Servers. By default Load Balancing Algorithms prefer Directory Servers in the same location as the Directory Proxy Server.

- Whether the Directory Server is degraded for any reason, such as having a Local DB Index being rebuilt.
- The result of configured Health Checks. For instance, a server with a small replication backlog can be preferred over one with a larger backlog.
- Recent operation routing history. For instance, the Round Robin Load Balancing Algorithm will sequentially route requests to all available servers. Typically Round Robin Load Balancing Algorithm does not perform as well as the Fewest Operations Load Balancing Algorithm.

How these factors are used depends on the specific Load Balancing Algorithm. The two most commonly used Load Balancing Algorithms are the Failover Load Balancing Algorithm and the Fewest Operations Load Balancing Algorithm. These two algorithms are similar when determining which Directory Servers are the possible candidates for a specific operation. The algorithms use the same criteria to determine server availability and health, and by default they will prefer Directory Servers in the same location as the Directory Proxy Server. However, they differ in the criteria they use to choose between available servers.

The Failover Load Balancing Algorithm will send all operations to a single server until it is unavailable, and then it will send all operations to the next preferred server, and so on. This algorithm provides the most consistent view of the topology to clients because all clients (at least those in the same location as the Directory Proxy Server) will see the same, up-to-date view of the data, but it leaves unused capacity in the failover instances since most topologies include multiple Directory Server replicas within each data center.

On the other hand, the Fewest Operations Load Balancing Algorithm does the best job of efficiently distributing traffic among multiple servers since it chooses to send each operation to the server that has the fewest number of outstanding operations--that is, the server from the Directory Proxy Server's point of view that is the least busy. (Note: the Fewest Operations Load Balancing Algorithm routes traffic to the least loaded server, which in a lightly-loaded environment can result in an imbalance since the first server in the list of configured servers is more likely to receive a request.) This algorithm naturally routes to servers that are more responsive as well as limiting the impact of servers that have become unreachable. However, this implies that consecutive operations that depend on each other can be routed to different Directory Servers, which can cause issues for some types of clients:

- If two entries are added in quick succession where the first entry is the parent of the second in the LDAP hierarchy, then the addition of the child entry could fail if that operation is routed to a different Directory Server instance than the first ADD operation, and this happens within the replication latency.
- Some clients add or modify an entry and then immediately read the entry back from the server, expecting to see the updates reflected in the entry.

In these situations, it is desirable to configure the Directory Proxy Server to route dependent requests to the same server.

The server affinity feature (see *Configuring Server Affinity*) achieves this in some environments but not in all because the affinity is tracked independently by each Directory Proxy Server instance, and some clients send requests to multiple Directory Proxy Servers. It is common for a client to not connect to the Directory Proxy Servers directly but instead to connect through a network load balancer, which in turn opens connections to the Directory Proxy Servers. Each individual client connection will be established to a single Directory Proxy Server so that operations on that connection will be routed to the same Directory Proxy Server, and server

affinity configured within the Directory Proxy Server will ensure those operations will be routed to the same Directory Server. However, many clients establish a pool of connections that are reused across operations, and within this pool, connections will be established through the load balancer to different Directory Proxy Servers. Dependent operations sent on different connections could then be routed to different Directory Proxy Servers, and then on to different Directory Servers.

A Failover Load Balancing Algorithm addresses this issue by routing all requests to a single server, but that leaves unused search capacity on the other instances. A Criteria Based Load Balancing Algorithm enables the proxy to route certain types of requests (or requests from certain clients) using a different Load Balancing Algorithm than the default. For instance, all write operations (i.e., ADD, DELETE, MODIFY, and MODIFY-DN) could be routed using a Failover Load Balancing Algorithm, while all other operations (bind, search, and compare) use a Fewest Operations Load Balancing Algorithm. And in addition, if there are clients that are particularly sensitive to reading entries immediately after modifying them, additional Connection Criteria can be specified to all operations from those clients using the Failover Load Balancing Algorithm. Note that, routing all write requests to a single server in a location instead of evenly across servers does not limit the overall throughput of the system since all servers ultimately have to process all write operations either from the client directly or via replication.

Another benefit of using the Failover Load Balancing Algorithm for write operations is reducing replication conflicts. The UnboundID Directory Server follows the traditional LDAP replication model of eventual consistency. This provides very high availability for handling write traffic even in the presence of network partitions, but it can lead to replication conflicts. Replication conflicts involving modify operations can be automatically resolved, leaving the servers in a consistent state where each attribute on each entry reflects the most recent update to that attribute. However, conflicts involving ADD, DELETE, and MODIFY-DN operations cannot always be resolved automatically and can require manual involvement from an administrator. By routing all write operations (or at least ADD, DELETE, and MODIFY-DN operations) to a single server, replication conflicts can be avoided.

There are a few points to consider when using a Failover Load Balancing Algorithm:

- When using the Failover Load Balancing Algorithm in a configuration with multiple locations, the Load Balancing Algorithm will fail over between local instances before failing over to servers in a remote location. The list of servers in the `backend-server` configuration property of the Load Balancing Algorithm should be ordered such that preferred local servers should appear before failover local servers, but the relative order of servers in different locations is unimportant as the `preferred-failover-location` of the Directory Proxy Server's configuration is used to decide which remote location to fail over to. It is also advisable that the order of local servers match the `gateway-priority` configuration settings of the "Replication Server" configuration object on the Directory Server instances. This can reduce the WAN replication delay because the Directory Proxy Server will then prefer to send writes to the Directory Server with the WAN Gateway role, avoiding an extra hop to the remote locations.

- For Directory Proxy Server configurations that include multiple Proxying Request Processors, including Entry Balancing environments, each Proxying Request Processor should be updated to include its own Criteria Based Load Balancing Algorithm.

To Route Operations to a Single Server

The following example shows how to extend a Directory Proxy Server's configuration to use a Criteria Based Load Balancing Algorithm to route all write requests to a single server using a Failover Load Balancing Algorithm. The approach outlined here can easily be extended to support alternate criteria as well as more complex topologies using multiple locations or Entry Balancing.

This example uses a simple deployment of a Directory Proxy Server fronting three Directory Servers: ds1.example.com, ds2.example.com, and ds3.example.com.

Once these configurations changes are applied, the Directory Proxy Server will route all write operations to ds1.example.com as long as it is available and then to ds2.example.com if it is not, while routing other types of operations, such as searches and binds, to all three servers using the Fewest Operations Load Balancing Algorithm.

1. First, create a location.

```
dsconfig create-location --location-name Austin
```

2. Update the failover location for your server.

```
dsconfig set-location-prop --location-name Austin
```

3. Set the location as a global configuration property.

```
dsconfig set-global-configuration-prop --set location:Austin
```

4. Set up the health checks for each external server.

```
dsconfig create-ldap-health-check \
--check-name ds1.example.com:389_dc_example_dc_com-search-health-check \
--type search --set enabled:true --set base-dn:dc=example,dc=com

dsconfig create-ldap-health-check \
--check-name ds2.example.com:389_dc_example_dc_com-search-health-check \
--type search --set enabled:true --set base-dn:dc=example,dc=com

dsconfig create-ldap-health-check \
--check-name ds3.example.com:389_dc_example_dc_com-search-health-check \
--type search --set enabled:true --set base-dn:dc=example,dc=com
```

5. Create the external servers.

```
dsconfig create-external-server --server-name ds1.example.com:389 \
--type unboundid-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 unboundid-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 unboundid-ds \
--set server-host-name:ds3.example.com --set server-port:389 \
--set location:Austin --set "bind-dn:cn=Proxy User,cn=Root DNs,cn=config" \
```

```
--set password:AADOkveb0TtYR9xpkVrNgMtF \
--set health-check:ds3.example.com:389_dc_example_dc_com-search-health-check
```

6. Create a Load Balancing Algorithm for `dc=example,dc=com`.

```
dsconfig create-load-balancing-algorithm \
--algorithm-name dc_example_dc_com-fewest-operations \
--type fewest-operations --set  enabled:true \
--set backend-server:ds1.example.com:389 \
--set backend-server:ds2.example.com:389 \
--set backend-server:ds3.example.com:389
```

7. Create a Request Processor for `dc=example,dc=com`.

```
dsconfig create-request-processor \
--processor-name dc_example_dc_com-req-processor \
--type proxying \
--set load-balancing-algorithm:dc_example_dc_com-fewest-operations
```

8. Create a Subtree View for `dc=example,dc=com`.

```
dsconfig create-subtree-view \
--view-name dc_example_dc_com-view \
--set base-dn:dc=example,dc=com \
--set request-processor:dc_example_dc_com-req-processor
```

9. Update the client connection policy for dc=example,dc=com.

```
dsconfig set-client-connection-policy-prop \
--policy-name default \
--add subtree-view:dc_example_dc_com-view
```

10. Create a new Request Criteria object to match all write operations.

```
dsconfig create-request-criteria \
--criteria-name any-write \
--type simple --set "description:All Write Operations" \
--set operation-type:add --set operation-type:delete \
--set operation-type:modify --set operation-type:modify-dn
```

11. Create a new Failover Load Balancing Algorithm listing the servers that should be included. Note the order that the servers are listed here is the failover order between servers.

```
dsconfig create-load-balancing-algorithm \
--algorithm-name dc_example_dc_com-failover \
--type failover --set enabled:true \
--set backend-server:ds1.example.com:389 \
--set backend-server:ds2.example.com:389 \
--set backend-server:ds3.example.com:389
```

12. Tie the Request Criteria and the Failover Load Balancing Algorithm together into a Criteria Based Load Balancing Algorithm.

```
dsconfig create-criteria-based-load-balancing-algorithm \
--algorithm-name dc_example_dc_com-write-traffic-lba \
--set "description:Failover LBA For All Write Traffic" \
--set request-criteria:any-write \
--set load-balancing-algorithm:dc_example_dc_com-failover
```

13. Update the Proxying Request Processor to use the Criteria Based Load Balancing Algorithm.

```
dsconfig set-request-processor-prop \
--processor-name dc_example_dc_com-req-processor \
```

```
--set criteria-based-load-balancing-algorithm:dc_example_dc_com-write-traffic-lba
```

To Route Operations from a Single Client to a Specific Server

To create a type of server affinity, where all operations from a single client are routed to a specific server, you follow a similar process as in the previous use case. However, instead of request criteria, you configure connection criteria. These connection criteria identify clients that could be adversely affected by replication latency. These clients will use the Failover Load Balancing Algorithm rather than the default Fewest Operations Load Balancing Algorithm.

For example, an administrative tool includes a "delete user" function. If the application immediately re-queries the directory for an updated list of users, the just-deleted entry must not be included. To configure a criteria-based load balancing algorithm to support this use case, you must do the following:

- Create a failover load balancing algorithm that lists the same set of servers as the existing fewest operation load balancing algorithm.
- Create connection criteria that match the clients for whom you want to apply failover load balancing rather than fewest operations load balancing.
- Create a criteria-based load balancing algorithm that references the two configuration objects you created in the previous steps.
- Create a criteria-based load balancing algorithm that references the two configuration objects you created in the previous steps.
- Assign the new load balancing algorithm to the proxying request processor.

The following procedure provides examples of each of these steps.

1. As in the previous section, you create the new failover load balancing algorithm using `dsconfig` as follows:

   ```
   dsconfig create-load-balancing-algorithm \
   --algorithm-name dc_example_dc_com-failover \
   --type failover --set enabled:true \
   --set backend-server:east1.example.com:389 \
   --set backend-server:east2.example.com:389 |
   --set backend-server:west1.example.com:389 \
   --set backend-server:west2.example.com:389
   ```

2. To route operations from a specific client to a single server in a set of replicated servers, you create connection criteria. Create the new connection criteria using `dsconfig` as follows:

   ```
   dsconfig create-connection-criteria \
   --criteria-name "Client One"  --type simple \
   --set included-user-base-dn:cn=Client One,ou=Apps,dc=example,dc=com
   ```

3. Configure a criteria-based load balancing algorithm and assign it to the proxying request processor. Use the load balancing algorithm and connection criteria you created in the previous steps to create the criteria-based load balancing algorithm:

   ```
   dsconfig create-criteria-based-load-balancing-algorithm \
   --algorithm-name dc_example_dc_com-client-operations \
   --set load-balancing-algorithm:dc_example_dc_com-failover \
   --set "request-criteria:Client One Requests"
   ```

4. Assign the new criteria-based load balancing algorithm to the proxying request processor using `dsconfig` as follows:

```
dsconfig set-request-processor-prop \
--processor-name dc_example_dc_com-req-processor \
--add criteria-based-load-balancing-agorithm:dc_example_dc_com-client-operations
```

Understanding Failover and Recovery

Once a previously degraded or unavailable server has recovered, it should be eligible to start receiving traffic within the time configured for the health-check-frequency property, 30 seconds by default. However, failover and recovery also depend on the load-balancing algorithm in use.

The load-balancing algorithm provides an ordered list of servers to check, with the number of servers in the list based on the maximum number of retry attempts. The server checks to see if affinity should be used and, if so, whether an affinity is set for that load-balancing algorithm. If there is an affinity to a particular server and that server is classified as available, then that server will always be the first in the list.

Next, the Directory Proxy Server creates a two-dimensional matrix of servers based on the health check state (with available preferred over degraded and unavailable not considered at all) and location (with backend servers in the same location as the Directory Proxy Server most preferred, then servers in the first failover location, then the second, and so on). Within each of these sets, and ideally at least one server in the local data center is classified as available, the load-balancing algorithm selects the servers in the order of most preferred to least preferred based on whatever logic the load-balancing algorithm uses. The load-balancing algorithm keeps selecting servers until enough of them have been selected to satisfy the maximum number of possible retries.

The load-balancing algorithm includes a configuration option that allows you to decide whether to prefer location over availability and vice versa. For example, is a local degraded server more or less preferred than a remote available server? By default, the algorithm will prefer available servers over degraded ones, even if it has to go to another data center to access them. You can change the load-balancing algorithm to try to stay in the same data center if at least one server is not unavailable.

The Directory Proxy Server does both proactive and reactive health checking. With proactive health checking, the Directory Proxy Server will periodically (by default, every 30 seconds), run a full set of tests against each backend server. The result of these tests will be usd to determine the overall health check state (available, degraded, or unavailable) and score (and integer value from 10 to 0). With reactive health checking, the Directory Proxy Server may kick off a lesser set of health checks against a server if an operation forwarded to that server did not complete successfully.

Proactive health checking can be used to promote and demote the health of a server, but reactive health checking can only be used to demote the health of a server. As a result, if a server is determined to be unavailable, then it will remain that way until a subsequence proactive health check determines that it has recovered. If a server is determined to be degraded, it may not become available until the next proactive health check, but it could be downgarded to unavailable by a reactive check if other failures are encountered against that server.

Both proactive and reactive health check assignments take effect immediately and will be considered for all subsequent requests routed to the load-balancing algorithm. If a a server is considered degraded, then it will immediately be considered less desirable than available servers in the same data center, and possibly less desirable than available servers in more remote data centers. If a server is considered unavailable, then it will not be eligible to be selected until it is reclassified as available or degraded.

Configuring Proxy Transformations

The UnboundID® Directory Proxy Server provides proxy transformations to alter the contents of client requests as they are sent from the client to the LDAP external server. Proxy transformations can also be used to alter the responses sent back from the server to the client, including altering or omitting search result entries. The Directory Proxy Server provides the following types of data transformations:

- **Attribute mapping**. This transformation rewrites client requests so that references to one attribute type may be replaced with an alternate attribute type. The proxy server can perform extensive replacements, including attribute names used in DNs and attribute names encoded in the values of a number of different controls and extended operations. For example, a client requests a `userid` attribute, which is replaced with `uid` before being forwarded on to the backend server. This mapping applies in reverse for the response returned to the client.

- **Default value**. This transformation instructs the directory proxy server to include a static attribute value in search results being sent back to the client, in ADD requests being forwarded to an external server, or both. For example, a value of "marketing" for `businessCategory` could be returned for all search results under the base DN `ou=marketing,dc=example,dc=com`.

- **DN mapping**. This transformation rewrites client requests so that references to entries below a specified DN will be mapped to appear below another DN. For example, references to entries below `o=example.com` could be rewritten so that they are below `dc=example,dc=com` instead. The mapping applies in reverse for the response returned to the client.

- **Groovy scripted**. This custom transformation is written in Groovy and does not need to be compiled, though they use the Server SDK. These scripts make it possible to alter requests and responses in ways not available using the transformations provided with the directory proxy server.

- **Suppress attribute**. This proxy transformation allows you to exclude a specified attribute from search result entries. It also provides the ability to reject add, compare, modify, modify DN, or search requests if they attempt to reference the target attribute.

- **Suppress entry**. This proxy transformation allows you to exclude any entries that match a specified filter from a set of search results. Search requests are transformed so that the original filter will be ANDed with a NOT filter containing the exclude filter. For example, if the suppression filter is `"(objectClass=secretEntry)"`, then a search request with a filter of `"(uid=john.doe)"` will be transformed so that it has a filter of `"(&(uid=john.doe)(!(objectClass=secretEntry)))"`.

- **Simple to external bind**. This proxy transformation may be used to intercept a simple bind request and instead process the bind as a SASL EXTERNAL bind. If the SASL EXTERNAL

bind fails, then the original simple bind request may or may not be processed, depending on how you configure the server.

- **Third-party scripted**. This custom transformation is created using the Server SDK, making it possible to alter requests and responses in ways not available using the transformations provided with the directory proxy server.

To Configure Proxy Transformations Using dsconfig

1. Use the `dsconfig` tool to create and configure a proxy transformation.

```
$ bin/dsconfig
```

2. Enter the connection parameters (for example, hostname, port, bind DN and bind DN password).

3. In the Directory Proxy Server configuration console menu, enter the number associated with proxy transformations. On the Proxy Transformation mangement menu, enter the number to create a new proxy transformation.

```
>>>> Proxy Transformation management menu

What would you like to do?

  1) List existing Proxy Transformations
  2) Create a new Proxy Transformation
  3) View and edit an existing Proxy Transformation
  4) Delete an existing Proxy Transformation

  b) back
  q) quit

Enter choice [1]: 2
```

4. Select the type of proxy transformation you want to create. In this example, we create an attribute mapping transformation. Then, enter a name for the new transformation.

```
 >>>> Enter a name for the Attribute Mapping Proxy Transformation that you
want to create: userid-to-uid
```

5. Indicate whether you want the transformation to be enabled by default.

```
Select a value for the 'enabled' property:

  1) true
  2) false

  ?) help
  c) cancel
  q) quit

Enter choice [c]: 1
```

6. Specify the name of the client attribute that you want to remap to a target attribute. Note that this attribute must not be equal to the target attribute.

```
Enter a value for the 'source-attribute' property: userid
```

7. Specify the name of the target attribute to which the client attribute should be mapped.

```
Enter a value for the 'target-attribute' property: uid
```

8. The properties of your new proxy transformation are displayed. If you want to make any further modifications, enter the number corresponding to the property. Enter `f` to finish the creation of the proxy transformation.

   ```
   Enter choice [b]: f
   ```

   The transformation now needs to be assigned to a request processor. To create an initial request processor, see the next section.

Configuring Request Processors

A request processor is responsible for handling client requests by passing the request through a load balancing algorithm or one or more subordinate request processors. The request processor is also the directory proxy server component that performs proxy transformations. You can create one of the following types of request processors:

- **Proxying request processor**. This request processor is responsible for passing allowed operations through a load balancing algorithm. Proxy transformations can be applied to requests and responses that are processed. Multiple servers may be configured to provide high availability and load balancing, and various transformations may be applied to the requests and responses that are processed.
- **Entry-balancing request processor**. This request processor is used to distribute entries under a common parent entry among multiple backend sets. A backend set is a collection of replicated directory servers that contain identical portions of the data. This request processor uses multiple, subordinate proxying request processors to process operations and maintains in-memory indexes to speed the processing of specific search and modify operations.
- **Failover request processor**. This request processor performs ordered failover between subordinate proxying processors, sometimes with different behavior for different types of operations. For example, you could use a failover request processor to achieve round-robin load balancing for read operations but failover load-balancing for writes.

To Configure Request Processors Using dsconfig

1. Use the `dsconfig` tool to create and configure a request processor.

   ```
   $ bin/dsconfig
   ```

2. Specify the hostname, connection method, port number, and bind DN as described in previous procedures.

3. In the Directory Proxy Server configuration console menu, enter the number associated with request processor configuration.

   ```
   Enter choice: 14

   >>>> Request Processor management menu
   ```

```
What would you like to do?

  1) List existing Request Processors
  2) Create a new Request Processor
  3) View and edit an existing Request Processor
  4) Delete an existing Request Processor

  b) back
  q) quit

Enter choice [b]: 2
```

4. Select an existing request processor to use as a template for creating a new one or enter `n` to create one from scratch. In this example, we create a new proxying request processor from scratch. You will be required to choose an existing load balancing algorithm or create a new one to complete the create of the request processor. Below is the configuration of the proxying request processor after selection of the load balancing algorithm.

```
Property                     Value(s)
-------------------------------------------------------------
1)  description              -
2)  enabled                  true
3)  allowed-operation        abandon, add, bind, compare, delete,
                             extended, modify, modify-dn, search
4)  load-balancing-algorithm dc_example_dc_com-fewest-operations
5)  transformation           -
6)  referral-behavior        pass-through
7)  supported-control        account-usable, assertion,
                             authorization-identity,
                             get-authorization-entry,
                             get-effective-rights, get-server-id,
                             ignore-no-user-modification,
                             intermediate-client, manage-dsa-it,
                             matched-values, no-op, password-policy,
                             permissive-modify, post-read, pre-read,
                             proxied-authorization-v1,
                             proxied-authorization-v2,
                             real-attributes-only, retain-identity,
                             subentries, subtree-delete,
                             virtual-attributes-only
8)  supported-control-oid    -

?)  help
f)  finish - create the new Proxying Request Processor
d)  display the equivalent dsconfig arguments to create this object
b)  back
q)  quit
```

5. Review the configuration properties of the new request processor. If you are satisfied, enter `f` to finish. For the request processor to be used, it must be associated with a subtree view.

To Pass LDAP Controls with the Proxying Request Processor

If your deployment does not use entry balancing and includes applications that use LDAP controls, such as the virtual list view (VLV) or LDAP join controls, you need to configure the directory proxy server to forward these controls correctly. You do so by configuring the `supported-control-oid` property to define the request OID of the LDAP control. The proxy updates the root DSE `supportedControl` attribute with the values entered for the `supported-control-oid` property. The request controls use the following OIDs:

- LDAP Join Control: 1.3.6.1.4.1.30221.2.5.9
- Virtual List View 2.16.840.1.113730.3.4.9
- Server-Side Sort: 1.2.840.113556.1.4.473
- Simple Paged Results: 1.2.840.113556.1.4.319

- Batched Transaction Specification Control: 1.3.6.1.4.1.30221.2.5.1
- Join Request Control: 1.3.6.1.4.1.30221.2.5.9

1. Use `dsconfig` to modify the `supported-control-oid` property. From the Request Processor management menu, enter the number corresponding to view and editing an existing request processor.

```
>>>> >>>> Request Processor management menu

What would you like to do?

    1)  List existing Request Processors
    2)  Create a new Request Processor
    3)  View and edit an existing Request Processor
    4)  Delete an existing Request Processor

    b)  back
    q)  quit

Enter choice [b]: 3
```

2. Enter the number corresponding to the `supported-control-oid` property.

3. Next, enter `2` to add a new value to the `supported-control-oid` property.

```
Do you want to modify the 'supported-control-oid' property?

    1)  Leave undefined
    2)  Add one or more values

    ?)  help
    q)  quit

Enter choice [1]: 2
```

4. Enter the request OID of the VLV control.

```
Enter a value for the 'supported-control-oid' property [continue]:
 2.16.840.1.113730.3.4.9
```

5. Enter `1` to use the `supported-control-oid` property value.

```
The 'supported-control-oid' property has the following values:

*)  2.16.840.1.113730.3.4.9

Do you want to modify the 'supported-control-oid' property?

    1)  Use these values
    2)  Add one or more values
    3)  Remove one or more values
    4)  Leave undefined
    5)  Revert changes

    ?)  help
    q)  quit

Enter choice [1]:
```

Configuring Server Affinity

The Directory Proxy Server supports the ability to forward a sequence of requests to the same external server if specific conditions are met. This feature, called server affinity, is applied by the load balancing algorithms. The following server affinity methods are available in the directory proxy server:

- **Client Connection**. Requests from the same directory proxy server client connection are consistently routed to the same external server.
- **Client IP**. Directory proxy server client requests coming from the same client IP address are routed to the same external server.
- **Bind DN**. Requests from all client connections authenticated as the same bind DN are routed to the same external server.

For each algorithm, you can specify the set of operations for which an affinity will be established, as well as the set of operations for which affinity will be used. Affinity assignments have a time-out value so that they are in effect for some period of time after the last operation that may cause the affinity to be set or updated.

To Configure Server Affinity

In this example, we create a bind DN server affinity provider for any client requesting write operations to have subsequent requests, whether read or write, forwarded to the same external server. The affinity period will last for 30 seconds after the last write request.

1. Use the `dsconfig` tool to configure server affinity. Specify the hostname, connection method, port number, and bind DN as described in previous procedures.

   ```
   $ bin/dsconfig
   ```

2. In the Directory Proxy Server configuration console menu, enter the number associated with server affinity provider configuration
3. On the Server Affinity management menu, enter the number corresponding to creating a new server affinity provider.
4. Enter a name for your new server affinity provider.

   ```
   >>>> Enter a name for the Bind DN Server Affinity Provider
   that you want to create: Affinity for Writing Applications
   ```

5. Indicate whether you want the server affinity provider to be enabled for use by the directory proxy server. In this example, enter `1` to enable to the server affinity provider.
6. Next, configure the properties of the server affinity provider. For example, you can customize the types of operations for which affinity may be set and the types of operations

for which affinity may be used, as well as the length of time for which the affinity should persist. This example illustrates the properties of the bind DN server affinity provider.

```
>>>> >>>> Configure the properties of the Bind DN Server Affinity Provider

        Property                 Value(s)
        -----------------------------------------------------------
    1)  description              -
    2)  enable                   true
    3)  affinity-duration        30 s
    4)  set-affinity-operation   add, delete, modify, modify-dn
    5)  use-affinity-operation   add, bind, compare, delete, modify,
                                 modify-dn, search

    ?)  help
    f)  finish - create the new Bind DN Server Affinity Provider
    d)  display the equivalent dsconfig arguments to create this object
    b)  back
    q)  quit

Enter choice [b]:
```

7. Review the properties of the server affinity provider. If you are satisfied, enter `f` to finish. Once defined, the affinity provider can now be assigned to a load balancing algorithm.

Configuring Subtree Views

You provide clients access to a specific portion of the DIT creating a subtree view and assigning it to a client connection policy. You can configure subtree views from the command line or using the Directory Proxy Server Management Console.

When you create a subtree view, you provide the following information to configure its properties:

- Subtree view name
- Base DN managed by the subtree view
- Request processor used by the subtree view to route requests. If one does not exist already, you will create a new one.

To Configure Subtree View

1. Use `dsconfig` to configure a subtree view.
2. In the Directory Proxy Server configuration console menu, enter the number associated with subtree view configuration
3. In the Subtree View management menu, enter the number corresponding to creating a new subtree view.
4. Enter a name for the subtree view.
5. Enter the base DN of the subtree managed by this subtree view.

```
Enter a value for the ‘base-dn’ property:dc=example,dc=com
```

6. Select a request processor for this subtree view to route requests or make the appropriate selection to create a new one.

```
Select a Request Processor for the ‘request-processor’ property:

1) dc_example_dc_com-req-processor
2) Create a new Request Processor

?) help
c) cancel
q) quit

Enter choice [c]: 1
```

7. Review the properties of the subtree view. If you are satisfied, enter `f` to finish.

```
>>>> Configure the properties of the Subtree View
 >>>> via creating 'example.com' Subtree View

        Property           Value(s)
        ---------------------------------------------------
    1)  description        -
    2)  base-dn            "dc=example,dc=com"
    3)  request-processor  dc_example_dc_com-req-processor

    ?)  help
    f)  finish - create the new Subtree View
    d)  display the equivalent dsconfig arguments to create
        this object
    b)  back
    q)  quit
```

Once configured, you can assign one or more subtree views to any client connection policies.

Configuring Client Connection Policies

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

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

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

Once a client has been assigned to a policy, you can determine what operations they can perform. For example, your policy might allow only SASL bind operations. Client connection

policies are also associated with one or more subtree views, which determine the portions of the DIT a particular client can access. For example, you might configure a policy that prevents users connecting over the extranet from accessing configuration information. The client connection policy is evaluated in addition to access control, so even a root user connecting over the extranet would not have access to the configuration information.

Understanding the Client Connection Policy

Client connection policies are based on two things:

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

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

When a Client Connection Policy is Assigned

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

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

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

All other types of operations will be subject to whatever constraints are defined for the client connection policy used by the client connection at the time that the request is received. The

client connection policy assigned to a connection never changes as a result of processing any operation other than a bind or StartTLS. So, the server will not re-evaluate the client connection policy for the connection in the course of processing an operation. For example, the client connection policy will never be re-evaluated for a search operation.

Restricting the Type of Search Filter Used by Clients

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

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

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

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

Defining Request Criteria

The client connection policy provides several properties that allow you to define the kinds of requests that it can issue. The `required-operation-request-criteria` property causes the server to reject any requests that do not match the referenced request criteria. The `prohibited-operation-request-criteria` property causes the server to reject any request that matches the referenced request criteria.

Setting Resource Limits

A client connection policy can specify resource limits, helping to ensure that no single client monopolizes server resources. The resource limits are applied in addition to any global configuration resource limits. In other words, a client connection policy cannot grant additional

resources beyond what is set in the global configuration. If a client connection exceeds either a globally-defined limit or a policy limit, then it is terminated.

Note: The directory proxy server's global configuration can enforce limits on the number of concurrent connections that can be established in the following ways:

- Limit the total number of concurrent connections to the server.
- Limit the total number of concurrent connections from the same IP address.
- Limit the total number of concurrent connections authenticated as the same bind DN.

Defining the Operation Rate

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

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

For example, the following are valid rate limit strings:

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

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

Client Connection Policy Deployment Example

In this example scenario, we assume the following:

- Two external LDAP clients are allowed to bind to the Directory Proxy Server.
- Client 1 should be allowed to open only 1 connection to the server.

- Client 2 should be allowed to open up to 5 connections to the server.

Defining the Connection Policies

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

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

Therefore, we define the following three client connection policies:

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

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

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

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

How the Policy is Evaluated

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

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

checks the connection criteria associated with the Client 2 Connection Policy because it has the second lowest evaluation order index. If these criteria do not match, the server finally checks the connection criteria associated with the Unauthenticated Connection Policy, because it has the third lowest evaluation order index. It finds a match, so the client connection is associated with the Unauthenticated Connection Policy.

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

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

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

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

To Configure a Client Connection Policy Using dsconfig

1. Use the `dsconfig` tool to create and configure a client connection policy.

   ```
   $ bin/dsconfig
   ```

2. Enter the connection parameters to the server (for example, hostname, connection method, port, bind DN and bind DN password).

3. In the Directory Proxy Server configuration console menu, enter the number associated with client connection policy configuration. Then enter the number to create a new client connection policy.

```
>>>> Client connection policy management menu

What would you like to do?

  1) List existing client connection policies
  2) Create a new client connection policy
  3) View and edit an existing client connection policy
  4) Delete an existing client connection policy

  b) back
  q) quit

Enter choice [b]: 2
```

4. Enter `n` to create a new client connection policy from scratch.

```
>>>> Select an existing Client Connection Policy to use as a
template for the new Client Connection Policy configuration or
'n' to create one from scratch:

  1) default

  n) new Client Connection Policy created from scratch
  c) cancel
  q) quit
```

5. Enter a name for the new client connection policy.

```
Enter the 'policy-id' for the Client Connection Policy that you
want to create: new_policy
```

6. Indicate whether you want the policy to be enabled by default.

```
Select a value for the 'enabled' property:

  1) true
  2) false

  ?) help
  c) cancel
  q) quit

Enter choice [c]: 1
```

7. Provide a value for the `evaluation-order-index` property. Client connection policies with a lower index will be evaluated before those with a higher index.

```
Enter a value for the 'evaluation-order-index' property: 2
```

8. The properties of your new client connection policy are displayed. If you want to make any further modifications, enter the number corresponding to the property. Enter `f` to finish the creation of the client connection policy.

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

Configuring Entry Balancing

With entry balancing, entries below a common parent, or balancing point, are split among multiple sets of directory backend servers. For information about configuring entry balancing with the `create-initial-proxy-config` tool, see *To Configure an Entry Balancing Directory Proxy Server Deployment*.

The remainder of this section describes special configuration considerations when planning for and setting up an entry balancing directory proxy server deployment. If you plan to use access control with your entry balancing deployment, refer to *Configuring Access Control with Entry Balancing*.

Determining How to Balance Your Data

If a single directory server instance can hold all of your data, then we recommend storing your data on a single server and replicating for high availability, as this simplifies your deployment. If a single server cannot hold all of your data, then you can spread it across multiple servers in several ways:

- If the data is already broken up by hierarchy and all of the clients understand how to access it that way, the number of top-level branches is small and a single directory server instance can hold all of the information within one or more branches. Configure the directory proxy server with multiple base DNs and use simple load-balancing rather than entry balancing to simplify your deployment.
- If simply breaking up the data using the existing hierarchy is not an option, for example if a large number of top-level branches must be configured, then consider using entry balancing. The contents of any single branch still must fit on a given server, because only entries that are immediate subordinates of the entry-balancing base DN may be spread across multiple servers. Any entries that are further subordinates have to be placed in the same directory instance as their parent.
- If one or more branches are so large that any single directory server instance cannot hold all of the data, you need to use entry balancing within that branch to divide the entries among two or more sets of directory servers. You may also need to change the way that the data is arranged in the server so that it uses as flat a DIT as possible, which is easier to use in an entry-balancing deployment.

In an entry-balancing deployment, there can be data that is common to all external directory servers outside the balancing point. This data is referred to as the global domain. A directory proxy server entry balancing configuration will contain at least two subtree views and associated request processors, one for the global domain and one for the entry balancing domain. In our examples, the global domain is `dc=example,dc=com` and the entry balancing domain is `ou=people, dc=example,dc=com`. The entry balancing base DN, `ou=people,dc=example,dc=com`, is also the balancing point.

Configuring an Entry Balancing Placement Algorithm

When receiving a client ADD request, the directory proxy server uses an entry-balancing placement algorithm to determine which backend server it forwards the request to. You can configure one of the following placement algorithms:

- **Entry counter placement algorithm**. This algorithm selects the backend set with the smallest number of entries or the smallest database size to distribute entries as equally amongst the backend servers as possible. This algorithm should be used for entry rebalancing.
- **Hash DN placement algorithm**. This algorithm forwards LDAP add requests to backend sets based on an MD5 hash of the entry DN. This algorithm ensures that a given DN maps repeatedly to the same backend set, provided that the backend sets do not change.
- **Round-robin placement algorithm**. This algorithm forwards LDAP add requests to backends sets in a round-robin manner.
- **Single set placement algorithm**. This algorithm forwards LDAP add requests to the same backend set.

Configuring Entry Rebalancing

If your deployment distributes entries using an entry counter placement algorithm or 3rd party algorithm, you may need to redistribute your entries. For example, imagine that you have an environment that distributes entries across three backends using an entry counter placement algorithm. This algorithm distributes entries to the backend that has the most space. Imagine that the backends all reach their maximum capacity and you decide to add a new backend to the deployment. You need to move the entries from the full backends and distribute them evenly across all the backends, including the new backend.

You might also want to deliberately rebalance your entries to meet the needs of your organization. For example, you can direct entry balancing based on attributes on the entries themselves. You can write a custom algorithm that looks at the value of an attribute that is being modified on the entry. Based on the attribute, you can then put this entry somewhere specific. You might use this feature if you want to have certain entries closer geographically to the client application using them. The geographical information could be included in the entry. Rebalancing would be used to move these entries to the server in the correct geographical location.

You can redistribute entry-balanced entries in two ways:

- **Using dynamic rebalancing.** With dynamic rebalancing, as existing entries get modified, they get moved. You configure dynamic rebalancing in the entry counter placement algorithm.

- **Using the `move-subtree` tool.** This tool can be used to move either small subtrees through a transactional method or to move large subtrees, potentially taking them offline for a short period.

The remainder of this section describes each of these method of entry rebalancing in more detail.

About Dynamic Rebalancing

During dynamic rebalancing entries get moved as they are modified. You configure dynamic rebalancing on the entry counter placement algorithm or a third-party placement algorithm that supports rebalancing. This algorithm keeps a count of the number of entries or the size of the backend set. You configure dynamic rebalancing using the following parameters:

- **`rebalancing-enabled`** - this parameter 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`** - this parameter indicates which modified entries are candidates for rebalancing. A value of top-level indicates that only entries immediately below the entry balancing base can be rebalanced. A value of any indicates that entries at any level below the entry balancing base may be rebalanced.
- **`rebalancing-minimum-percentage`** - this parameter 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.

The following figure illustrates an entry balancing base DN and three subtrees, A, B, and C. If the rebalancing scope is set to `any`, any child entries under the base DN can be rebalanced. For example, if a change is made to entry A1, the entire subtree A might be rebalanced, depending upon how you have configured rebalancing. If the rebalancing scope is set to `top-level`, rebalancing is only triggered when entries at the top level, such as A, are modified. Changes made to subentries, such as A1 or A2, do not trigger rebalancing. Rebalancing is also triggered upon the addition of entries such as `A1,A2`, provided the scope is `any`.

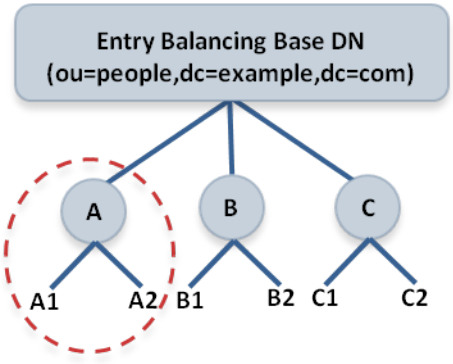


Figure 3: Rebalancing at the Top Level

If you are writing your own 3rd party algorithm, you program dynamic rebalancing using the `SelectRebalancingBackendSet` method on the placement algorithm. For more information, see the server SDK documentation.

To Configure Dynamic Rebalancing

This procedure describes how to configure dynamic rebalancing on an existing entry balancing configuration.

1. To configure entry rebalancing, you may create an entry counter placement algorithm, if the current placement algorithm does not support rebalancing. You can either do this using `dsconfig` in interactive mode, or using the `dsconfig` command line as follows:

```
$ dsconfig create-placement-algorithm \
                --processor-name dc_example_dc_com-eb-req-processor \
                --algorithm-name rebalancing --type entry-counter \
                --set enabled:true --set rebalancing-enabled:true
```

2. Remove any placement algorithm previously configured on this entry balancing request processor.

3. You can throttle how many entries are being moved by the proxy, so that the backend servers do not have too heavy a load. To do this, set the rebalancing-queue-maximum-size property of the request processor created in the previous step. By default, it is set to 1000. If the load is too high, reduce this value as follows:

```
$ dsconfig set-request-processor-prop \
                     --processor-name dc_example_dc_com-eb-req-processor \
                --set rebalancing-queue-maximum-size:50
```

4. Verify that the access logs are configured to display the subtrees being moved by dynamic rebalancing. The access logs provide a good way to monitor progress. So, if the write load on the backend servers is high and you see lots of rebalancing activity in the access log, lower the queue size to lower the rebalancing activity. You can configure the access log to display entry rebalancing processing information as follows:

```
$ dsconfig set-log-publisher-prop \
                --publisher-name "File-Based Access Logger" \
                --set log-entry-rebalancing-requests:true
```

About the `move-subtree` Tool

The `move-subtree` tool allows you to specify subtrees for rebalancing. You specify the source server, the target server, and one or more base DNs identifying the subtrees you want to move. You can move small subtrees using the transactional method or move large subtrees, which does not use this method. Instead, the large subtree is not fully accessible during the move, so clients may get an "insufficient access rights error" if they try to access the subtree. As entries are moved, clients can read but not write to them. Once the transfer is complete, the entries are fully available to client requests.

This tool accepts a file containing a list of the base DNs of the subtrees you want to move.

About the `subtree-accessibility` Tool

The `subtree-accessibility` tool help you evaluate if a subtree has restricted access and helps you fix any problems. If, during rebalancing, the Directory Server issues an alert that a subtree has been unavailable for too long, then you can use this tool to evaluate the problem. For example, if the `move-subtree` tool is interrupted by a host machine going down unexpectedly, the subtree might not be successfully moved. You can use the `subtree-accessibility` tool to evaluate and correct any problems with the subtrees.

Chapter 4 Managing the Directory Proxy Server

Once you have configured the UnboundID® Directory Proxy Server, you can manage the day-to-day operations of your deployment using the monitoring and logging features. This chapter provides procedures to help you configure logging and monitor your deployment.

This chapter includes the following sections:

Topics:

Managing Logs

The Directory Proxy Server provides a number of different types of log publishers that can be used to provide information about how the server is processing.

About the Default Logs

You can view all logs in the `UnboundID-Proxy/logs` directory. This section provides information about the following default logs:

- Error Log
- server.out Log
- Debug Log
- Config Audit Log and the Configuration Archive
- Access Log
- Setup Log
- Tool Log
- LDAP SDK Debug Log

Error Log

By default, this log file is available at `logs/errors` below the server install root and it provides information about warnings, errors, and other significant events that occur within the server. A number of messages are written to this file on startup and shutdown, but while the server is running there is normally little information written to it. In the event that a problem does occur, however, the server writes information about that problem to this file.

The following is an example of a message that might be written to the error log:

```
[11/Apr/2011:10:31:53.783 -0500] category=CORE severity=NOTICE msgID=458887 msg="The
 Directory Server has started successfully"
```

The category field provides information about the area of the server from which the message was generated. Available categories include:

ACCESS_CONTROL, ADMIN, ADMIN_TOOL, BACKEND, CONFIG, CORE, DSCONFIG, EXTENSIONS, PROTOCOL, SCHEMA, JEB, SYNC, LOG, PLUGIN, PROXY, QUICKSETUP, REPLICATION, RUNTIME_INFORMATION, TASK, THIRD_PARTY, TOOLS, USER_DEFINED, UTIL, VERSION.

The severity field provides information about how severe the server considers the problem to be. Available severities include:

- **DEBUG** – Used for messages that provide verbose debugging information and do not indicate any kind of problem. Note that this severity level is rarely used for error logging, as the Directory Proxy Server provides a separate debug logging facility as described below.

- **INFORMATION** – Used for informational messages that can be useful from time to time but are not normally something that administrators need to see.
- **MILD_WARNING** – Used for problems that the server detects, which can indicate something unusual occurred, but the warning does not prevent the server from completing the task it was working on. These warnings are not normally something that should be of concern to administrators.
- **MILD_ERROR** – Used for problems detected by the server that prevented it from completing some processing normally but that are not considered to be a significant problem requiring administrative action.
- **NOTICE** – Used for information messages about significant events that occur within the server and are considered important enough to warrant making available to administrators under normal conditions.
- **SEVERE_WARNING** – Used for problems that the server detects that might lead to bigger problems in the future and should be addressed by administrators.
- **SEVERE_ERROR** – Used for significant problems that have prevented the server from successfully completing processing and are considered important.
- **FATAL_ERROR** – Used for critical problems that arise which might leave the server unable to continue processing operations normally.

The messages written to the error log may be filtered based on their severities in two ways. First, the error log publisher has a `default-severity` property, which may be used to specify the severity of messages logged regardless of their category. By default, this includes the NOTICE, SEVERE_WARNING, SEVERE_ERROR, and FATAL_ERROR severities.

You can override these severities on a per-category basis using the `override-severity` property. If this property is used, then each value should consist of a category name followed by an equal sign and a comma-delimited set of severities that should be logged for messages in that category. For example, the following override severity would enable logging at all severity levels in the PROTOCOL category:

```
protocol=debug,information,mild-warning,mild-error,notice,severe-warning,severe-
error,fatal-error
```

Note that for the purposes of this configuration property, any underscores in category or severity names should be replaced with dashes. Also, severities are not inherently hierarchical, so enabling the DEBUG severity for a category will not automatically enable logging at the INFORMATION, MILD_WARNING, or MILD_ERROR severities.

The error log configuration may be altered on the fly using tools like `dsconfig`, the web administration console, or the LDIF connection handler, and changes will take effect immediately. You can configure multiple error logs that are active in the server at the same time, writing to different log files with different configurations. For example, a new error logger may be activated with a different set of default severities to debug a short-term problem, and then that logger may be removed once the problem is resolved, so that the normal error log does not contain any of the more verbose information.

server.out Log

The `server.out` file holds any information written to standard output or standard error while the server is running. Normally, it includes a number of messages written at startup and shutdown, as well as information about any administrative alerts generated while the server is running. In most cases, this information is also written to the error log. The `server.out` file can also contain output generated by the JVM. For example, if garbage collection debugging is enabled, or if a stack trace is requested via "kill -QUIT" as described in a later section, then output is written to this file.

Debug Log

The debug log provides a means of obtaining information that can be used for troubleshooting problems but is not necessary or desirable to have available while the server is functioning normally. As a result, the debug log is disabled by default, but it can be enabled and configured at any time.

Some of the most notable configuration properties for the debug log publisher include:

- **enabled** – Indicates whether debug logging is enabled. By default, it is disabled.
- **log-file** – Specifies the path to the file to be written. By default, debug messages are written to the `logs/debug` file.
- **default-debug-level** – Specifies the minimum log level for debug messages that should be written. The default value is “error,” which only provides information about errors that occur during processing (for example, exception stack traces). Other supported debug levels include warning, info, and verbose. Note that unlike error log severities, the debug log levels are hierarchical. Configuring a specified debug level enables any debugging at any higher levels. For example, configuring the info debug level automatically enables the warning and error levels.
- **default-debug-category** – Specifies the categories for debug messages that should be written. Some of the most useful categories include caught (provides information and stack traces for any exceptions caught during processing), database-access (provides information about operations performed in the underlying database), protocol (provides information about ASN.1 and LDAP communication performed by the server), and data (provides information about raw data read from or written to clients).

As with the error and access logs, multiple debug loggers can be active in the server at any time with different configurations and log files to help isolate information that might be relevant to a particular problem.

Note: Enabling one or more debug loggers can have a significant impact on server performance. We recommend that debug loggers be enabled only when necessary, and then be scoped so that only pertinent debug information is recorded.

Debug targets can be used to further pare down the set of messages generated. For example, you can specify that the debug logs be generated only within a specific class or package. If you need to enable the debug logger, you should work with your authorized support provider to best configure the debug target and interpret the output.

Audit log

The audit log is a specialized version of the access log, used for troubleshooting problems that may occur in the course of processing. The log records all changes to directory data in LDIF format so that administrators can quickly diagnose the changes an application made to the data or replay the changes to another server for testing purposes.

The audit log does not record authentication attempts but can be used in conjunction with the access log to troubleshoot security-related issues. The audit log is disabled by default because it does adversely impact the server's write performance.

By default, if you enable the audit log on the Proxy Server, the userPassword and authPassword attribute values are obscured, Each value of an obscured attribute is replaced in the audit log with a string of the form `"***** OBSCURED VALUE *****"`. You can unobscure these attributes by deleting them from the `obscure-attribute` property.

Config Audit Log and the Configuration Archive

The configuration audit log provides a record of any changes made to the server configuration while the server is online. This information is written to the `logs/config-audit.log` file and provides information about the configuration change in the form that may be used to perform the operation in a non-interactive manner with the `dsconfig` command. Other information written for each change includes:

- Time that the configuration change was made.
- Connection ID and operation ID for the corresponding change, which can be used to correlate it with information in the access log.
- DN of the user requesting the configuration change and the method by which that user authenticated to the server.
- Source and destination addresses of the client connection.
- Command that can be used to undo the change and revert to the previous configuration for the associated configuration object.

In addition to information about the individual changes that are made to the configuration, the Directory Proxy Server maintains complete copies of all previous configurations. These configurations are provided in the `config/archived-configs` directory and are gzip-compressed copies of the `config/config.ldif` file in use before the configuration change was made. The filenames contain time stamps that indicate when that configuration was first used.

Access and Audit Log

The access log provides information about operations processed within the server. The default access log file is written to `logs/access`, but multiple access loggers can be active at the same time, each writing to different log files and using different configurations.

By default, a single access log message is generated, which combines the elements of request, forward, and result messages. If an error is encountered while attempting to process the request, then one or more forward-failed messages may also be generated.

```
[01/Jun/2011:11:10:19.692 -0500] CONNECT conn=49 from="127.0.0.1" to="127.0.0.1"
  protocol="LDAP+TLS" clientConnectionPolicy="default"
[01/Jun/2011:11:10:19.764 -0500] BIND RESULT conn=49 op=0 msgID=1 version="3"
  dn="cn=Directory Manager" authType="SIMPLE" resultCode=0 etime=0.401
  authDN="cn=Directory Manager,cn=Root DNs,cn=config" clientConnectionPolicy="default"
[01/Jun/2011:11:10:19.769 -0500] SEARCH RESULT conn=49 op=1 msgID=2
  base="ou=People,dc=example,dc=com" scope=2 filter="(uid=1)" attrs="ALL"
  resultCode=0 etime=0.549 entriesReturned=1
[01/Jun/2011:11:10:19.788 -0500] DISCONNECT conn=49 reason="Client Unbind"
```

Each log message includes a timestamp indicating when it was written, followed by the operation type, the connection ID (which is used for all operations processed on the same client connection), the operation ID (which can be used to correlate the request and response log messages for the operation), and the message ID used in LDAP messages for this operation.

The remaining content for access log messages varies based on the type of operation being processed, and whether it is a request or a result message. Request messages generally include the most pertinent information from the request, but generally omit information that is sensitive or not useful.

Result messages include a `resultCode` element that indicates whether the operation was successful or if failed and an `etime` element that indicates the length of time in milliseconds that the server spent processing the operation. Other elements that might be present include the following:

- **origin=replication** – Operation that was processed as a result of data synchronization (for example, replication) rather than a request received directly from a client.
- **message** – Text that was included in the `diagnosticMessage` field of the response sent to the client.
- **additionalInfo** – Additional information about the operation that was not included in the response sent back to the client.
- **authDN – DN** of the user that authenticated to the server (typically only included in bind result messages).
- **authzDN – DN** of an alternate authorization identify used when processing the operation (for example, if the proxied authorization control was included in the request).
- **authFailureID** – Unique identifier associated with the authentication failure reason (only included in non-successful bind result messages).

- **authFailureReason** – Information about the reason that a bind operation failed that might be useful to administrators but was not included in the response to the client for security reasons.
- **responseOID** – OID included in an extended response returned to the client.
- **entriesReturned** – Number of matching entries returned to the client for a search operation.
- **unindexed=true** – Indicates that the associated search operation could not be sufficiently processed using server indexes and a significant traversal through the database was required.

Note that this is not an exhaustive list, and elements that are not listed here may also be present in access log messages. The Commercial Edition of the LDAP SDK provides an API for parsing access log messages and provides access to all elements that they may contain.

The Directory Proxy Server provides a second access log implementation called the *audit log*, which is used to provide detailed information about write operations (add, delete, modify, and modify DN) processed within the server. If the audit log is enabled, the entire content of the change is written to the audit log file (which defaults to `logs/audit`) in LDIF form.

The UnboundID® Directory Proxy Server also provides a very rich classification system that can be used to filter the content for access log files. This can be helpful when debugging problems with client applications, because it can restrict log information to operations processed only by a particular application (for example, based on IP address and/or authentication DN), only failed operations, or only operations taking a long time to complete, etc.

Setup Log

The `setup` tool writes a log file providing information about the processing that it performs. By default, this log file is written to `logs/setup.log` although a different name may be used if a file with that name already exists, because the `setup` tool has already been run. The full path to the setup log file is provided when the `setup` tool has completed.

Tool Log

Many of the administrative tools provided with the Directory Proxy Server (for example, `import-ldif`, `export-ldif`, `backup`, `restore`, etc.) can take a significant length of time to complete write information to standard output or standard error or both while the tool is running. They also write additional output to files in the `logs/tools` directory (for example, **logs/ tools/ import-ldif.log**). The information written to these log files can be useful for diagnosing problems encountered while they were running. When running via the server tasks interface, log messages generated while the task is running may alternately be written to the server error log file.

LDAP SDK Debug Log

This log can be used to help examine the communication between the Directory Proxy Server and the Directory Proxy Server. It contains information about exceptions that occur during processing, problems establishing and terminating network connections, and problems that occur

during the reading and writing of LDAP messages and LDIF entries. You can configure the types of debugging that should be enabled, the debug level that should be used, and whether debug messages should include stack traces. As for other file-based loggers, you can also specify the rotation and retention policies.

Types of Log Publishers

The UnboundID® Directory Proxy Server provides a number of differently types of loggers that can be used to get processing information about the server. There are three primary types of loggers:

- **Access loggers** provide information about operations processed within the server. They can be used for understanding the operations performed by clients and debugging problems with directory-enabled applications, and they can also be used for collecting usage information for performance and capacity planning purposes.
- **Error loggers** provide information about warnings, errors, or significant events that occur within the server.
- **Debug loggers** can provide detailed information about processing performed by the server, including any exceptions caught during processing, detailed information about data read from or written to clients, and accesses to the underlying database.

By default, the following log publishers are enabled on the system:

- File-based access logger
- File-based error logger
- Failed-operations access logger

The UnboundID® Directory Proxy Server also provides the follow log publishers that are disabled by default:

- File-based debug logger
- File-based audit logger
- Expensive operations access logger
- Successful searches with no entries returned access logger

Creating New Log Publishers

The UnboundID® Directory Proxy Server provides customization options to help you create your own log publishers with the `dsconfig` command.

When you create a new log publisher, you must also configure the log retention and rotation policies for each new publisher. For more information, see Configuring Log Rotation and Configuring Log Retention.

To Create a New Log Publisher

1. Use the `dsconfig` command in non-interactive mode to create and configure the new log publisher. This example shows how to create a logger that only logs disconnect operations.

```
$ bin/dsconfig create-log-publisher \
  --type file-based-access --publisher-name "Disconnect Logger" \
  --set enabled:true \
  --set "rotation-policy:24 Hours Time Limit Rotation Policy" \
  --set "rotation-policy:Size Limit Rotation Policy" \
  --set "retention-policy:File Count Retention Policy" \
  --set log-connects:false \
  --set log-requests:false --set log-results:false \
  --set log-file:logs/disconnect.log
```

Note: To configure compression on the logger, add the option to the previous command:

```
--set compression-mechanism: gzip
```

Compression cannot be disabled or turned off once configured for the logger. Therefore, careful planning is required to determine your logging requirements including log rotation and retention with regards to compressed logs.

2. View the Log Publishers.

```
$ bin/dsconfig list-log-publishers
```

```
Log Publisher           : Type           : enabled
------------------------:------------------:--------
Disconnect Logger          : file-based-access : true
File-Based Access Logger   : file-based-access : true
File-Based Audit Logger    : file-based-access : false
File-Based Debug Logger    : file-based-debug  : false
File-Based Error Logger    : file-based-error  : true
Replication Repair Logger  : file-based-error  : true
```

To Create a Log Publisher Using dsconfig Interactive Command-Line Mode

1. On the command line, type `bin/dsconfig`.
2. Authenticate to the server by following the prompts.
3. On the UnboundID® Directory Proxy Server Configuration console main menu, select the option to configure the log publisher.
4. On the **Log Publisher Management** menu, select the option to create a new log publisher.
5. Select the Log Publisher type. In this case, select **File-Based Access Log Publisher**.
6. Type a name for the log publisher.
7. Enable it.

8. Type the path to the log file, relative to the Directory Proxy Server root. For example, `logs/disconnect.log`.
9. Select the rotation policy you want to use for your log publisher.
10. Select the retention policy you want to use for your log publisher.
11. On the Log Publisher Properties menu, select the option for `log-connects:false`, `log-disconnects:true`, `log-requests:false`, and `log-results:false`.
12. Type `f` to apply the changes.

Configuring Log Rotation

The Directory Proxy Server allows you to configure the log rotation policy for the server. When any rotation limit is reached, the Directory Proxy Server rotates the current log and starts a new log. If you create a new log publisher, you must configure at least one log rotation policy.

You can select the following properties:

- **Time Limit Rotation Policy**. Rotates the log based on the length of time since the last rotation. Default implementations are provided for rotation every 24 hours and every 7 days.
- **Fixed Time Rotation Policy**. Rotates the logs every day at a specified time (based on 24-hour time). The default time is 2359.
- **Size Limit Rotation Policy**. Rotates the logs when the file reaches the maximum size for each log. The default size limit is 100 MB.
- **Never Rotate Policy**. Used in a rare event that does not require log rotation.

To Configure the Log Rotation Policy

- Use `dsconfig` to modify the log rotation policy for the access logger.

```
$ bin/dsconfig set-log-publisher-prop \
  --publisher-name "File-Based Access Logger" \
  --remove "rotation-policy:24 Hours Time Limit Rotation Policy" \
  --add "rotation-policy:7 Days Time Limit Rotation Policy"
```

Configuring Log Retention

The Directory Proxy Server allows you to configure the log retention policy for each log on the server. When any retention limit is reached, the Directory Proxy Server removes the oldest archived log prior to creating a new log. Log retention is only effective if you have a log rotation policy in place. If you create a new log publisher, you must configure at least one log retention policy.

- **File Count Retention Policy**. Sets the number of log files you want the Directory Proxy Server to retain. The default file count is 10 logs. If the file count is set to 1, then the log will continue to grow indefinitely without being rotated.
- **Free Disk Space Retention Policy**. Sets the minimum amount of free disk space. The default free disk space is 500 MBytes.
- **Size Limit Retention Policy**. Sets the maximum size of the combined archived logs. The default size limit is 500 MBytes.
- **Custom Retention Policy**. Create a new retention policy that meets your Directory Proxy Server's requirements. This will require developing custom code to implement the desired log retention policy.
- **Never Delete Retention Policy**. Used in a rare event that does not require log deletion.

To Configure the Log Retention Policy

- Use `dsconfig` to modify the log retention policy for the access logger.

```
$ bin/dsconfig set-log-publisher-prop \
  --publisher-name "File-Based Access Logger" \
  --set "retention-policy:Free Disk Space Retention Policy"
```

Managing the Global Indexes in Entry Balancing Configurations

In an entry balancing configuration, the Directory Proxy Server maintains the default RDN index as well as one or more in-memory global attribute indexes. The global indexes allow the proxy server to select the correct backend server set for incoming operations, which avoids broadcasting operations to all backend sets.

The indexes may be preloaded from peer proxies or the backend directory servers when the server starts up, and are updated by certain operations that come through the proxy. For instance, when a new entry is added, the DN of the new entry is added to the DN index of the proxy server performing the operation. The indexes are also fault-tolerant and can adapt to changes made in the backend servers without going through the directory proxy server. For example, operations will be processed directly through the backend server.

This section describes when to create a global attribute index, how to reload the global index, how to monitor its growth, and how to prime the global index from a peer at start-up.

When to Create a Global Attribute Index

The RDN index is referenced whenever a modify, delete, or base search is requested. In other words, the RDN index is needed when the LDAP request contains the complete DN of the

targeted entry. If the entry balancing request processor is not configured to prime the rdn index at startup, then the index is populated over time as LDAP requests are processed.

A global attribute index is an optional index and is referenced when the proxy is handling a search request with an equality filter involving the attribute, such as the `telephoneNumber` attribute with the filter (telephoneNumber=+11234567890). Since the proxy server does not know what the data within the subtree views looks like or how it will be searched, it cannot create or recommend default global attribute index definitions. The creation of a global attribute index is based on the administrator's knowledge of the range of equality-filtered search requests the proxy will handle.

The common candidates for global attribute indexing are the uniquely-valued equality-indexed attributes on the external servers. Examples of these attributes are `uid`, `mail` and `telephoneNumber`. Though the values of the attribute need not be unique to be used as a global attribute index by the entry balancing request processor.

Consider a proxy deployment that expects to handle frequent searches of the form "`(&(mail=user@example.com)(objectclass=person))`". Since the filter is constructed with an equality match and &-clause, we can use a global attribute index on the `mail` attribute to avoid forwarding the search request to each entry balanced dataset.

The following `dsconfig` command creates the global attribute index. We are declaring the mail attribute to be uniquely valued across the entry balanced datasets. Note that the mail attribute must be indexed for equality searches on each of the external servers behind the proxy server.

```
$ bin/dsconfig create-global-attribute-index \
  --processor-name ou_people_dc_example_dc_com-eb-req-processor \
  --index-name mail --set prime-index:true \
  --set guaranteed-unique:true
```

After creating the index with `dsconfig`, the index will begin to be populated as search requests involving the mail attribute are made to the proxy. At this point, you can also use the `reload-index` tool to fully populate the index for optimal performance as described in the following section.

Reloading the Global Indexes

The Directory Proxy Server provides a tool, `reload-index`, which allows you to manually reload the directory proxy server global indexes. You might need to reload the index when:

- The proxy fails to successfully load its global indexes on startup.
- Changes are made to the set of indexed attributes.
- Significant changes are made to the content in backend servers.
- The integrity of the index is in question.

You can use the tool to reload all configured indexes in the global index, including the RDN index and all attribute indexes, or to reload only those indexes you specify.

The tool schedules an operation to run within the directory proxy server's process. You must supply LDAP connection information so that the tool can communicate with the server through

its task interface. Tasks can be scheduled to run immediately or at a later scheduled time. Once scheduled, you can manage the tasks using the `manage-tasks` tool.

To Reload All of the Index

- Run the `reload-index` tool to reload all of the indexes within the scope of the `dc=example,dc=com` base DN. The task is performed as `cn=Directory Manager` on port 389 of the localhost server. The existing index contents are erased before reloading.

```
$ bin/reload-index --task --bindPassword password --baseDN "dc=example,dc=com"
```

To Reload the RDN and UID Index

- To reload the RDN and UID index in the background so that the existing contents of these indexes can continue to be used, run the command as follows:

```
$ bin/reload-index --task --bindPassword password \
  --baseDN "dc=example,dc=com" --index rdn --index uid --background
```

To Prime the Backend Server Using the --fromDS Option

You can force the proxy server to prime from the backend directory server only using the `--fromDS` option, overriding the configuration of the `prime-index-source` property. You can do this on a one off basis if the global index appears to be growing too large. For example, run the command as follows:

- Run the `reload-index` command with the `--fromDS` option to prime the backend server.

```
$ bin/reload-index --bindPassword password --baseDN "dc=example,dc=com" --fromDS
```

Monitoring the Size of the Global Indexes

Over time, stale entries can build up in the global indexes because proxy servers do not communicate changes to the indexes with one another. The proxy continues to operate normally in this situation since the global indexes are only ever used as a hint at where to find entries.

The rate of this growth is typically very slow since in most environments the key attributes change infrequently. The global indexes themselves are also very compact. However, if the global indexes start to fill up the allocated memory, you may need to flush and reload them. The size of the global indexes can be monitored over LDAP using the following command:

```
$ bin/ldapsearch -b "cn=monitor" -D "uid=admin,dc=example,dc=com" -w password \
  "(objectClass=ds-entry-balancing-request-processor-monitor-entry)" \
  global-index-current-memory-percent
```

If the global indexes fill up, the proxy server will continue to operate normally, but it will need to start evicting entries from the indexes, which will lead to more broadcast searches, reducing the overall throughput of the proxy server.

To reload the indexes so that they no longer hold stale information, run the `reload-index` command with the `--fromDS` option so that data is loaded from backend directory servers. We

recommend that you reload the indexes during off-peak hours because it may have an impact on performance while the reload is in progress.

Sizing the Global Indexes

The directory proxy server includes a tool, `global-index-size`, to help you estimate the size in memory of your global indexes. You can estimate the size of more than one index in a single invocation by providing multiple sets of options. The tool creates its estimate using the following information:

- **Number of keys in the index**. For example, for the built-in RDN index, the number of keys is the total number of entries in the directory that are immediately below the balancing point. Entries more than one level below the balancing point, as well as entries that are not subordinate to the balancing point, will not be contained in the RDN index. For attribute indexes, the number of keys will be the number of unique values for that attribute in the entry-balanced portion of the data.
- **Average size of each key, in bytes**. For attributes indexes, the key is simply the attribute value. For the built-in RDN index, the key is the RDN directly below the balancing base DN. For example, for the DN `uid=user.0,dc=example,dc=com` under the balancing base DN of `dc=example,dc=com`, the key size is 10 bytes (the number of bytes in the RDN uid=user.0).
- **Estimated number of keys**. This value corresponds to the maximum number of keys you expect in your directory. The number of keys is provided in the index-size configuration property of the `global-attribute-index` object when you configure an attribute index. For the built-in RDN index, the configured number of keys is provided in the `rdn-index-size` property. If you do not provide a value, the tool assumes that the configured number of keys is the same as the actual number of keys.

To Size the Global Index

- Run the `global-index-size` to estimate the size of two separate indexes, both with 10,000,000 keys but with differing average key sizes. The configured number of keys is assumed to be equal to the actual number of keys:

```
$ bin/global-index-size --numKeys 10000000 \
  --averageKeySize 11 --numKeys 10000000 \
  --averageKeySize 15

Num Keys : Cfg. Num Keys : Avg. Key Size : Est. Memory Size
---------:---------------:---------------:-----------------
10000000 : 10000000      : 11            : 159 mb
10000000 : 10000000      : 15            : 197 mb
```

Priming the Global Indexes on Start Up

The Directory Proxy Server can prime the global indexes on startup from the backend directory server or from a peer directory proxy server, preferably one that resides on the same LAN or subnet. When priming occurs locally, you can avoid WAN bandwidth consumption and reduce the processing load on the directory servers in the topology. You can specify the data sources for the index priming and the order in which priming from these sources occurs.

Use the `prime-index-source` property to specify the sources of data, either `ds`, `proxy` or some combination of the two. The order you specify is the order in which priming from these sources will be attempted. For example, if you specify `prime-index-source:ds,proxy`, priming will first be attempted from backend directory servers, then from peer directory proxy servers. In most cases, your directory proxy server should prime its index from backend directory servers instead of from a peer proxy server, so that it gets the most up-to-date information. No matter the server you choose, priming is most efficient if the source server is on the same local network as the directory proxy server.

In some circumstances, such as when the proxy server fronts a Sun Directory Server (where priming from the directory server can be expensive and inefficient), you may prefer priming the indexes from a peer directory proxy server instance. The `peer-proxy-server` property allows you to specify the set of directory proxy servers that are available to process index primingrequests. The `prime-from-proxy-maximum-retry-count` property allows you to specify the times that priming is retried against an alternate peer proxy server. If no other peer directory proxy server instances are available, then the server may be configured to fail over to priming from the backend servers.

Root accounts and external servers are automatically configured when defining peer proxy servers in an entry-balancing topology. The root account created is called `cn=IntraProxy User,cn=Root DNs,cn=config`, and the external server name is `intra-proxy-<servername>`. If you have already configured peer proxy servers in a topology, use these existing intra-proxy accounts to specify the peer proxy server in the entry-balancing request processor.

To Prime All Indexes at Startup

You can change the entry-balancing request processor so that it loads all indexes at startup from the peer proxy server. If the peer is unavailable, the request processor loads the indexes from the backend directory servers.

- Run the `dsconfig` tool to prime all indexes at startup.

```
$ bin/dsconfig set-request-processor-prop \
  --processor-name dc_example_dc_com-eb-req-processor \
  --set prime-all-indexes:true --set prime-index-source:proxy \
  --set prime-index-source:ds \
  --add peer-proxy-server:intra-proxy-host.example.com:3389
```

Note: Setup creates an intra-proxy external server for the proxy server itself. This server should not be chosen as a peer server.

To Prime the Global Indexes Manually

If you do not want to configure priming during setup, you can configure index priming manually by creating an external server, creating a global attribute index, and then changing the entry-balancing request processor to load indexes from this external server.

1. Use the `dsconfig` tool to create an external server of the type UnboundID® Directory Proxy Server to represent a peer of the proxy server.

```
$ bin/dsconfig create-external-server \
  --server-name intra-proxy-host.example.com:3389 \
```

```
  --type UnboundID-Proxy-server \
  --set server-host-name:intra-proxy-host \
  --set server-port:338 \
  --set "bind-dn:cn=Directory Manager" \
  --set "password:secret123"
```

2. Next, create a global attribute index on the `uid` attribute as follows:

```
$ bin/dsconfig create-global-attribute-index \
  --processor-name dc_example+dc+com-eb-req-processor \
  --index-name uid --set guaranteed-unique:true
```

3. Finally, change the entry-balancing request processor to load the indexes at startup from the peer proxy server using `dsconfig set-request-processor-prop` as described above.

Priming or Reloading the Global Indexes from Sun Directory Servers

When priming or reloading a global index based on a Sun Directory Server environment, the Sun servers may become overwhelmed and unresponsive because of their method of streaming data. To reduce the impact of priming on these server, you can use the `prime-search-entry-per-second` property. To reduce the impact of reloading these indexes, use the `--searchEntryPerSecond` property of the `reload-index` command. These properties control the rate at which the proxy server accepts search result entries from the backend directory servers.

To Prime or Reload the Global Indexes

The following example configures the entry-balancing request processor so that it loads all indexes at startup from the peer proxy server. If the peer is unavailable, the request processor loads the indexes from the backend Sun directory server at a rate of 1000 entries per second.

1. Run the `dsconfig` command to set the entry-balancing request processor to load all indexes at startup.

```
$ bin/dsconfig set-request-processor-prop \
  --processor-name dc_example_dc_com-eb-req-processor \
  --set prime-all-indexes:true --set prime-index-source:proxy \
  --set prime-index-source:ds \
  --add peer-proxy-server:intra-proxy-host.example.com:2389 \
  --set prime-search-entry-per-second 1000
```

2. To control the rate at which the proxy accepts search result entries from the backend directory server during index reload, you set the `--searchEntryPerSecond` property as follows:

```
$ bin/reload-index --bindPassword password --baseDN "dc=example,dc=com" \
  --searchEntryPerSecond 1000
```

3. To find the optimum rate, we recommend starting low, for example, specifying a few thousand search entries per second, and then increasing as necessary.

Setting Resource Limits

You can set resource limits for the Directory Proxy Server using several global configuration properties as well as setting resource limits on specific client connection policies. If you configure both global and client connection policy resource limits, the first limit reached will always be honored. For example, if the server-wide maximum concurrent connections limit is reached, then all subsequent connection will be rejected until existing connections are closed, regardless of whether a client connection policy limit has been reached.

Setting Global Resource Limits

You can specify the following types of global resource limits:

- Specify the maximum number of client connections that can be established at any given time using the `maximum-concurrent-connections` property. If the server already has the maximum number of connections established, then any new connection attempts from any clients will be rejected until an existing connection is closed. The default value of zero indicates that no limit is enforced.
- Specify the maximum number of client connections that can be established at any give time from the same client system using the `maximum-concurrent-connections-per-ip-address` property. If the server already has the maximum number of connections established from a given client, then any new connection attempts from that client will be rejected until an existing connection from that client is closed. The server may continue to accept connections from other clients that have not yet reached this limit. The default value of zero indicates that no limit is enforced.
- Specify the maximum number of client connections that can be established at any given time while authenticated as a particular user with the `maximum-concurrent-connections-per-bind-dn` property. This property applies after the connection is established, because the bind operation to authenticate the user happens after the connection is established rather than during the course of establishing the connection itself. If the maximum number of connections are authenticated as a given user, then any new attempt to authenticate as that user will cause the connection performing the bind to be terminated. Note that this limit applies only to authenticated connections, and will not be enforced for clients that have not authenticated or for clients that have authenticated as the anonymous user. The default value of zero indicates that no limit is enforced.

Any changes to the `maximum-concurrent-connections` and `maximum-concurrent-connections-per-ip-address` properties will take effect only for new connections established after the change is made. Any change to the `maximum-concurrent-connections-per-bind-dn` property will apply only to connections (including existing connections) which perform authentication after the change is made. Existing connections will be allowed to remain established even if that would cause the new limit to be exceeded.

Setting Client Connection Policy Resource Limits

You can also configure resource limits in a client connection policy using the following properties of the client connection policy:

- **maximum-concurrent-connections**. This property specifies the maximum number of client connections that may be associated with a specific client connection policy at any given time. Once this limit has been reached, any further attempts to associate a connection with this client connection policy will result in the termination of the connection.
- **maximum-connection-duration**. This property specifies the maximum length of time that a connection associated with a particular client connection policy may be established. When the connection has been established longer than this period, it will be terminated.
- **maximum-idle-connection-duration**. This property specifies the maximum time that a connection associated with a particular client connection policy may remain established after the completion of the last operation processed on that connection. Any new operation requested on the connection resets the timer. Connections that are idle for longer than the specified time will be terminated.
- **maximum-operation-count-per-connection**. This property specifies the maximum number of operations that may be requested by any client connection associated with this client connection policy. If an attempt is made to process more than this number of operations on the connection, then the connection will be terminated.
- **maximum-concurrent-operations-per-connection**. This property specifies the maximum number of concurrent operations that can be in progress for any connection. This property can be used to prevent a single client connection from monopolizing server processing resources by sending a large number of concurrent asynchronous requests.
- **maximum-connection-operation-rate**. This property specifies the maximum rate at which a client associated with a specific client connection policy may issue requests to the directory proxy server. If a client attempts to request operations at a rate higher than this limit, then the server will behave as described by the `connection-operation-rate-exceeded-behavior` property.
- **connection-operation-rate-exceeded-behavior**. This property describes how the server should behave if a client connection attempts to exceed a rate defined in the `maximum-connection-operation-rate` property.
- **maximum-policy-operation-rate**. This property specifies the maximum rate at which all clients associated with a particular client connection policy may issue requests to the directory proxy server. If this limit is exceeded, then the server will exhibit the behavior described in the `policy-operation-rate-exceeded-behavior` property.
- **policy-operation-rate-exceeded-behavior**. This property specifies the behavior of the directory proxy server if a client connection attempts to exceed the rate defined in the `maximum-policy-operation-rate` property.

Monitoring the Directory Proxy Server

While the Directory Proxy Server is running, it generates a significant amount of information available through monitor entries. This section contains information about the following:

- Monitoring Server Status Using the status Tool
- About the Monitor Entries
- Using the Monitoring Interfaces
- Monitoring with JMX

To Monitor Server Using the Status Tool

The UnboundID® Directory Proxy Server provides a `status` tool that provides basic server status information, including version, connection handlers, a table of LDAP external servers, and the percent of the global index that is used.

1. Run the `status` tool to view the current state of the server.

```
$ bin/status
```

2. Enter the LDAP connection parameters.

```
>>>> Specify LDAP connection parameters

Administrator user bind DN [cn=Directory Manager]:

Password for user 'cn=Directory Manager':
```

```
          --- Server Status ---
Server Run Status:    Started 07/Jan/2011:10:59:52.000 -0600
Operational Status:   Available
Open Connections:     4
Max Connections:      8
Total Connections:    25

          --- Server Details ---
Host Name:            example
Administrative Users: cn=Directory Manager
Installation Path:    /path/to/UnboundID-Proxy
Version:              UnboundID® Directory Proxy Server 3.6.0.0
Java Version:         Sun/Oracle JDK 1.6.0_31

          --- Connection Handlers ---

Address:Port : Protocol : State
-------------:----------:---------

0.0.0.0:1689 : JMX      : Disabled
0.0.0.0:636  : LDAPS    : Disabled
0.0.0.0:9389 : LDAP     : Enabled

         --- LDAP External Servers ---

Server         : Status    : Score : LB Algorithm
---------------:-----------:-------:----------------------
localhost:389  : Available : 10    : dc_example_dc_com-round-robin
localhost:1389 : Available : 10    : dc_example_dc_com-round-robin
```

```
         --- LDAP External Server Op Counts ---

Server         : Add : Bind:Compare:Delete:Modify:Mod DN:Search : All
---------------:-----:----:------:-------:-------:------:-------:----
localhost:11389: 0   : 0  : 0    : 0     : 0     : 0    : 1249  : 1249
localhost:12389: 0   : 0  : 0    : 0     : 0     : 0    : 494   : 494


         --- Entry Balancing Request Processors ---

 Base DN                     : Global Index % Used
-----------------------------:--------------------
ou=people,dc=example,dc=com : 33


         --- Global Index Stats for ou=people,dc=example,dc=com ---

Index : Total Bytes : Key Bytes : Keys : Size (# Keys) : Inserted :
Removed : Replaced: Hits : Misses : Discarded : Duplicates
------:-------------:-----------:---------:---------------:----------
rdn   : 30667304    : 14888906  : 1000001 : 3464494 0 :0 :0 :0 :0 :0
uid   : 26523480    : 10888902  : 1000001 : 3464494 0 : 0 : 3583 : 0 : 0 : 0


          --- Operation Processing Time ---

Op Type   : Total Ops : Avg Resp Time (ms)
----------:-----------:-------------------
Add       : 0         : 0.0
Bind      : 0         : 0.0
Compare   : 0         : 0.0
Delete    : 0         : 0.0
Modify    : 0         : 0.0
Modify DN : 0         : 0.0
Search    : 3583      : 117.58
All       : 3583      : 117.58


          --- Work Queue ---

           : Recent : Average : Maximum
-----------:--------:---------:--------
Queue Size : 0      : 0       : 1
% Busy     : 0      : 1       : 19
```

About the Monitor Entries

While the Directory Proxy Server is running, it generates a significant amount of information available through monitor entries. Monitor entries are available over LDAP in the `cn=monitor` subtree. The types of monitor entries that are available include:

- **General Monitor Entry (cn=monitor)** – Provides a basic set of general information about the server.
- **Active Operations Monitor Entry (cn=Active Operations,cn=monitor)** – Provides information about all operations currently in progress in the server.
- **Backend Monitor Entries (cn={id} Backend,cn=monitor)** – Provides information about the backend, including the number of entries, the base DN(s), and whether it is private.
- **Client Connections Monitor Entry (cn=Client Connections,cn=monitor)** – Provides information about all connections currently established to the server.
- **Connection Handler Monitor Entry (cn={name},cn=monitor)** – Provides information about the configuration of each connection handler and the client connections established to it.

- **Database Environment Monitor Entries (cn={id} Database Environment,cn=monitor)** – Provides statistics and other data from the Oracle Berkeley DB Java Edition database 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.
- **System Information Monitor Entry (cn=System Information,cn=monitor)** – Provides information about the underlying JVM and system.
- **Version Monitor Entry (cn=Version,cn=monitor)** – Provides information about the Directory Proxy Server version.
- **Work Queue Monitor Entry (cn=Work Queue,cn=monitor)** – Provides information about the state of the Directory Proxy Server work queue, including the number of operations waiting on worker threads and the number of operations that have been rejected because the queue became full.

Monitoring System Data Using the Metrics Engine

The UnboundID Metrics Engine provides collection and storage of performance data from your server topology. You can use the System Utilization Monitor with the Metrics Engine to collect information about the host system CPU, disk, and network utilization on any platform except Linux. If you are not using the Metrics Engine, you do not need to use the system utilization monitor. When data is being collected, it periodically forks the process and executes commands.

For more information about using the System Utilization Monitor, refer to the data collection chapter of the UnboundID Metrics Engine documentation.

Using the Monitoring Interfaces

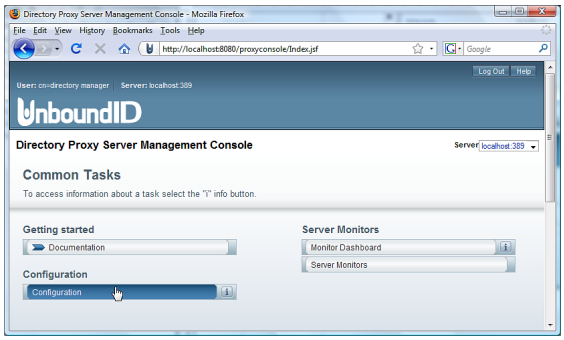
The UnboundID® Directory Proxy Server exposes its monitoring information under the `cn=monitor` entry and provides interfaces through the Directory Proxy Management Console, JMX, over LDAP, using the LDAP SDK, and using SNMP.

Monitoring with the Directory Management Console

UnboundID has developed a graphical web console for administrators to configure the directory server. The console also provides a monitoring component that accesses the server's monitor content.

To View the Monitor Dashboard

1. Ensure that the Directory Server is running.
2. Open a browser to `http://hostname:8080/dsconsole/`. For information about installing the Directory Server Management Console, see Installing the Directory Management Console.
3. Type the root user DN (or any authorized administrator user name) and password, and then click **Login**.
4. Click **Monitor Dashboard**.



5. View the monitoring information on the dashboard.

Accessing the Processing Time Histogram

The UnboundID® Directory Proxy Server provides a processing time histogram that classifies operation response time into user-defined buckets. The histogram tracks the processing on a per operation basis and as a percentage of the overall processing time for all operations. It also provides statistics for each operation type (add, bind, compare, delete, modify, modify DN, search).

To Access the Processing Time Histogram

1. On the **Directory Management Console**, click **Server Monitors**.
2. Click **Processing Time Histogram**. Other monitor entries can be accessed in similar ways.

Monitoring with JMX

The UnboundID® Directory Proxy Server supports monitoring the JVM™ through a Java Management Extensions (JMX™) management agent, which can be accessed using JConsole or any other kind of JMX client. The JMX interface provides JVM performance and resource utilization information for applications running Java. You can monitor generic metrics exposed by the JVM itself, including memory pools, threads, loaded classes, and MBeans, as well as all the monitor information that the Directory Proxy Server provides. You can also subscribe to receive JMX notifications for any administrative alerts that are generated within the server.

Running JConsole

Before you can access JConsole, you must configure and enable the JMX Connection Handler for the Directory Proxy Server using the `dsconfig` tool. See Configuring the JMX Connection Handler and Alert Handler.

To invoke the JConsole executable, type `jconsole` on the command line. If *JDK_HOME* is not set in your path, you can access JConsole in the `bin` directory of the `JDK_HOME` path.

To Run JConsole

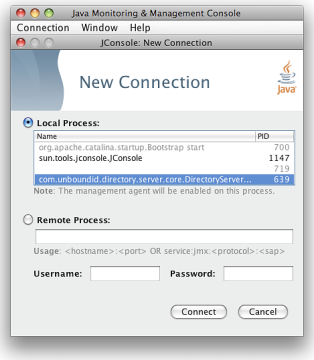
1. Use `JConsole` to open the Java Monitoring and Management Console. You can also run JConsole to monitor a specific process ID for your application: `jconsole PID`. Or you can run JConsole remotely using: `jconsole hostname:port`.

```
$ jconsole
```

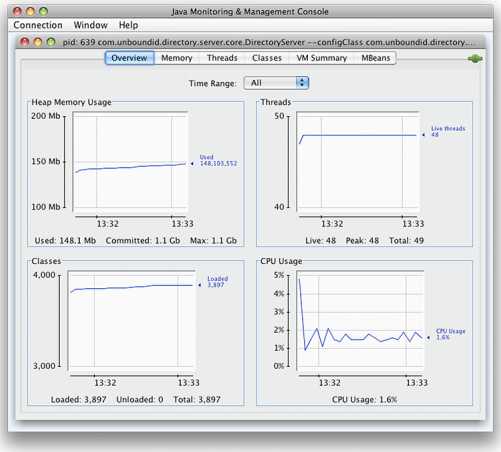
Note: If SSL is configured on the JMX Connection Handler, you must specify the Directory Server jar file in the class path when running `jconsole` over SSL. For example, run the following `jconsole` command:

```
$ jconsole \
  -J-Djavax.net.ssl.trustStore=/path/to/certStores/truststore \
  -J-Djavax.net.ssl.trustStorePassword=secret \
  -J-Djava.class.path=$SERVER_ROOT/UnboundID-Proxy.jar
```

2. On the **Java Monitoring & Management Console**, click **Local Process**, and then click the **PID** corresponding to the directory server.



3. Review the resource monitoring information.



Monitoring the Directory Proxy Server Using JConsole

You can set up JConsole to monitor the Directory Proxy Server using a remote process. Make sure to enable the JMX Connection Handler and to assign at least the `jmx-read` privilege to a regular user account (the `jmx-notify` privilege is required to subscibe to receive JMX notifications). Do not use a root user account, as this would pose a security risk.

To Monitor the Directory Server using JConsole

1. Start the Directory Proxy Server.

```
$ bin/start-ds
```

2. Enable the JMX Connection handler using the `dsconfig` tool. The handler is disabled by default. Remember to include the LDAP connection parameters (hostname, port, bindDN, bindPassword).

```
$ bin/dsconfig set-connection-handler-prop \
  --handler-name "JMX Connection Handler" --set enabled:true
```

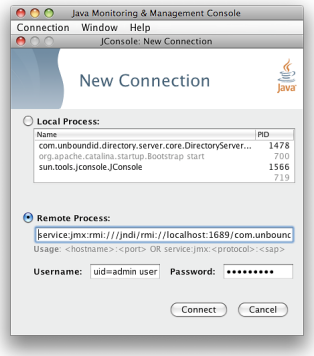
3. Assign `jmx-read`, `jmx-write`, and `jmx-notify` (if the user receives notifications) to the user.

```
$ bin/ldapmodify --hostname server1.example.com --port 1389 \
  --bindDN "cn=Directory Manager" --bindPassword secret
dn: uid=admin,dc=example,dc=com
changetype: modify
replace: ds-privilege-name
ds-privilege-name: jmx-read
ds-privilege-name: jmx-write
ds-privilege-name: jmx-notify
```

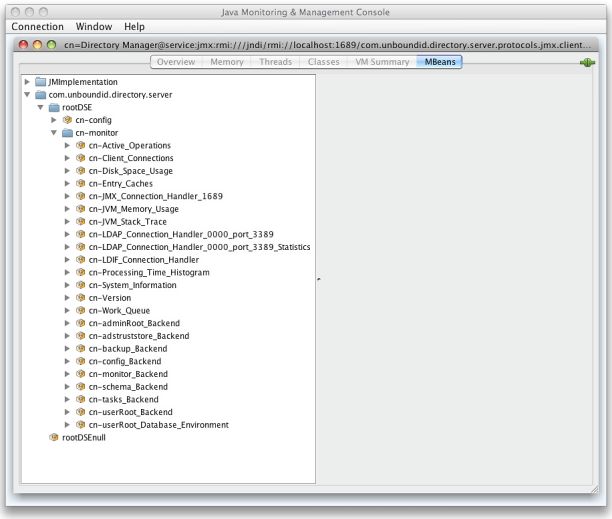
4. On the **Java Monitoring & Management Console**, click **Remote Process**, and enter the following JMX URL using the host and port of your directory.

```
service:jmx:rmi:///jndi/rmi://<host>:<port>/
com.unboundid.directory.server.protocols.jmx.client-unknown
```

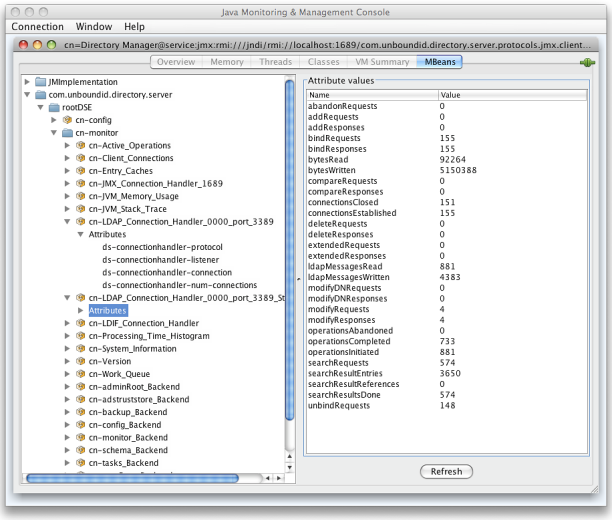
5. In the **Username** and **Password** fields, type the bind DN and password for a user that has at least the `jmx-read` privilege. Click **Connect**.



6. Click **com.unboundid.directory.server**, and expand the `rootDSE` node and the `cn-monitor` sub-node.



7. Click a monitoring entry. In this example, click the **LDAP Connection Handler** entry.



Monitoring over LDAP

The UnboundID® Directory Proxy Server exposes a majority of directory information under the `cn=monitor` entry. You can access these entries over LDAP using the `ldapsearch` tool.

```
$ bin/ldapsearch --hostname server1.example.com --port 1389 \
  --bindDN "uid=admin,dc=example,dc=com" --bindPassword secret \
  --baseDN "cn=monitor" "(objectclass=*)"
```

Monitoring Using the LDAP SDK

You can use the monitoring API to retrieve monitor entries from the directory proxy server as well as to retrieve specific types of monitor entries.

For example, you can retrieve all monitor entries published by the Directory Proxy Server and print the information contained in each using the generic API for accessing monitor entry data as follows:

```
for (MonitorEntry e : MonitorManager.getMonitorEntries(connection))
  {
    System.out.println("Monitor Name: " + e.getMonitorName());
    System.out.println("Monitor Type: " + e.getMonitorDisplayName());
    System.out.println("Monitor Data:");
    for (MonitorAttribute a : e.getMonitorAttributes().values())
    {
      for (Object value : a.getValues())
      {
        System.out.println(" " + a.getDisplayName() + ": " + String.valueOf(value));
      }
     }
     System.out.println();
  }
```

For more information about the LDAP SDK and the methods in this example, see the *UnboundID LDAP SDK* documentation.

Monitoring Using SNMP

The UnboundID® Directory Proxy Server supports real-time monitoring using the Simple Network Management Protocol (SNMP). The Directory Proxy Server provides an embedded SNMPv3 subagent plugin that, when enabled, sets up the server as a managed device and exchanges monitoring information with a master agent based on the AgentX protocol.

SNMP Implementation

In a typical SNMP deployment, many production environments use a network management system (NMS) for a unified monitoring and administrative view of all SNMP-enabled devices. The NMS communicates with a master agent, whose main responsibility is to translate the SNMP protocol messages and multiplex any request messages to the subagent on each managed device (for example, Directory Server instance, Directory Proxy Server, Synchronization Server, or OS Subagent). The master agent also processes responses or traps from the agents. Many vendors provide commercial NMS systems, such as Alcatel-Lucent (Omnivista EMS), HP (OpenView), IBM-Tivoli (Netview), Oracle-Sun (Solstice Enterprise Manager), and others. Specific discussion on integrating an SNMP deployment on an NMS system is beyond the scope of this chapter. Consult with your NMS system for specific information.

The UnboundID® Directory Proxy Server contains an SNMP subagent plug-in that connects to a Net-SNMP master agent over TCP. The main configuration properties of the plug-in are the address and port of the master agent, which default to localhost and port 705, respectively. When the plug-in is initialized, it creates an AgentX subagent and a managed object server, and then registers as a MIB server with the Directory Proxy Server instance. Once the plug-in's startup method is called, it starts a session thread with the master agent. Whenever the connection is lost, the subagent automatically attempts to reconnect with the master agent. The Directory Proxy Server’s SNMP subagent plug-in only transmits read-only values for polling or trap purposes (set and inform operations are not supported). SNMP management applications cannot perform actions on the server on their own or by means of an NMS system.

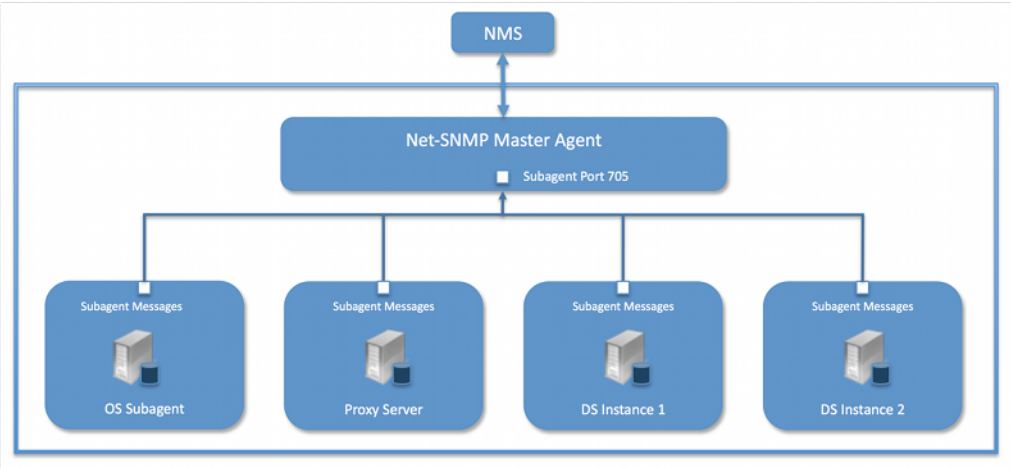


Figure 4: Example SNMP Deployment

One important note is that the UnboundID® Directory Proxy Server was designed to interface with a Net-SNMP (version 5.3.2.2 or later) master agent implementation with AgentX over TCP. Many operating systems provide their own Net-SNMP module, such as the System Management Agent (SMA) on Solaris or OpenSolaris. However, SMA disables some features present in the Net-SNMP package and only enables AgentX over UNIX Domain Sockets, which cannot be supported by Java. If your operating system has a native Net-SNMP master agent that only enables UNIX Domain Sockets, you must download and install a separate Net-SNMP binary from its web site.

Configuring SNMP

Because all server instances provide information for a common set of MIBs, each server instance provides its information under a unique SNMPv3 context name, equal to the server instance name. The server instance name is defined in the Global Configuration, and is constructed from the host name and the server LDAP port by default. Consequently, information must be requested using SNMPv3, specifying the context name that pertains to the desired server instance. This context name is limited to 30 characters or less. Any context name longer than 30 characters will result in an error message. Since the default context name is limited to 30 characters or less, and defaults to the server instance name and the LDAP port number, pay special attention to the length of the fully-qualified (DNS) hostname.

Note: The Directory Proxy Server supports SNMPv3, and only SNMPv3 can access the MIBs. For systems that implement SNMP v1 and v2c, Net-SNMP provides a proxy function to route requests in one version of SNMP to an agent using a different SNMP version.

To Configure SNMP

1. Enable the Directory Proxy Server's SNMP plug-in using the `dsconfig` tool. Make sure to specify the address and port of the SNMP master agent. On each Directory Proxy Server instance, enable the SNMP subagent. Note that the SNMPv3 context name is limited to 30 bytes maximum. If the default dynamically-constructed instance name is greater than 30 bytes, there will be an error when attempting to enable the plugin.

```
$ bin/dsconfig set-plugin-prop --plugin-name "SNMP Subagent" \
  --set enabled:true --set agentx-address:localhost \
  --set agentx-port:705 --set session-timeout:5s \
  --set connect-retry-max-wait:10s
```

2. Enable the SNMP Subagent Alert Handler so that the sub-agent will send traps for administrative alerts generated by the server.

```
$ bin/dsconfig set-alert-handler-prop \
  --handler-name "SNMP Subagent Alert Handler" --set enabled:true
```

3. View the error log. You will see a message that the master agent is not connected, because it is not yet online.

```
The SNMP sub-agent was unable to connect to the master
agent at localhost/705: Timeout
```

4. Edit the SNMP agent configuration file, `snmpd.conf`, which is often located in `/etc/snmp/snmpd.conf`. Add the directive to run the agent as an AgentX master agent:

   ```
   master agentx agentXSocket tcp:localhost:705
   ```

   Note that the use of `localhost` means that only sub-agents running on the same host can connect to the master agent. This requirement is necessary since there are no security mechanisms in the AgentX protocol.

5. Add the trap directive to send SNMPv2 traps to `localhost` with the community name, public (or whatever SNMP community has been configured for your environment) and the port.

   ```
   trap2sink localhost public 162
   ```

6. To create a SNMPv3 user, add the following lines to the `/etc/snmp/snmpd.conf` file.

   ```
   rwuser initial
   createUser initial MD5 setup_passphrase DES
   ```

7. Run the following command to create the SNMPv3 user.

   ```
   snmpusm -v3 -u initial -n "" -l authNoPriv -a MD5 -A setup_passphrase \
   localhost create snmpuser initial
   ```

8. Start the `snmpd` daemon and after a few seconds you should see the following message in the Directory Proxy Server error log:

   ```
   The SNMP subagent connected successfully to the master agent
   at localhost:705. The SNMP context name is host.example.com:389
   ```

9. Set up a trap client to see the alerts that are generated by the Directory Proxy Server. Create a config file in `/tmp/snmptrapd.conf` and add the directive below to it. The directive specifies that the trap client can process traps using the public community string, and can log and trigger executable actions.

   ```
   authcommunity log, execute public
   ```

10. Install the MIB definitions for the Net-SNMP client tools, usually located in the `/usr/share/snmp/mibs` directory.

    ```
    $ cp resource/mib/* /usr/share/snmp/mibs
    ```

11. Then, run the trap client using the `snmptrapd` command. The following example specifies that the command should not create a new process using `fork()` from the calling shell (`-f`), do not read any configuration files (`-C`) except the one specified with the `-c` option, print to standard output (`-Lo`), and then specify that debugging output should be turned on for the User-based Security Module (`-Dusm`). The path after the `-M` option is a directory that contains the MIBs shipped with our product (i.e., `server-root/resource/mib`) .

    ```
    $ snmptrapd -f -C -c /tmp/snmptrapd.conf -Lf /root/trap.log -Dusm \
      -m all -M +/usr/share/snmp/mibs
    ```

12. Run the Net-SNMP client tools to test the feature. The following options are required: `-v <SNMP version>`, `-u <user name>`, `-A <user password>`, `-l <security level>`, `-n <context name (instance name)>` . The `-m` all option loads all MIBs in the default MIB

directory in `/usr/share/snmp/mibs` so that MIB names can be used in place of numeric OIDs.

```
$ snmpget -v 3 -u snmpuser -A password -l authNoPriv -n host.example.com:389 \
-m all localhost localDBBackendCount.0

$ snmpwalk -v 3 -u snmpuser -A password -l authNoPriv -n host.example.com:389 \
-m all localhost systemStatus
```

13. If you want alerts sent from the SNMP Subagent through the Net-SNMP master agent and onwards, you must enable the SNMP Subagent Alert Handler. The SNMP Alert Handler is used in deployments that do not enable the Subagent.

```
$ bin/dsconfig --no-prompt 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
```

Configuring SNMP on AIX

Native AIX SNMP implementations do not support AgentX sub-agents, which is a requirement for UnboundID® Directory Proxy Servers. To implement SNMP on AIX platforms, any freely-available net-snmp package must be installed.

Special care must be made to ensure that you are using the net-snmp binary packages and not the native snmp implementation. Third-party net-snmp binary packages typically install under `/opt/freeware` and have the following differences:

```
Native Daemon: /usr/sbin/snmpd
Native Configuration File: /etc/snmpd.conf, /etc/snmpdv3.conf
Native Daemon Start and Stop: startsrc -s snmpd, stopsrc -s snmpd

net-snmp Daemon: /opt/freeware/sbin/snmpd
net-snmp Configuration File: /opt/freeware/etc/snmp/snmpd.conf
net-snmp start and stop: /etc/rc.d/init.d/snmpd start|stop
```

When configuring an SNMP implementation on AIX, remember to check the following items so that the Directory Proxy Server is referencing the net-snmp installation:

- The shell PATH will reference the native implementation binaries. Adjust the PATH variable or invoke the net-snmp binaries explicitly.
- If the native daemon is not stopped, there will likely be port conflicts between the native daemon and the net-snmp daemon. Disable the native daemon or use distinct port numbers for each.

SNMP on AIX Security Considerations

On AgentX sub-agent-compliant systems, it is recommended to use `agentXSocket tcp:localhost:705` to configure the net-snmp master agent to allow connections only from sub-agents located on the same host. On AIX systems, it is possible to specify an external IP network interface (for example, `agentXSocket tcp:0.0.0.0:708` would listen on all external IP interfaces), which would allow the UnboundID® Directory Proxy Server to be located on a different host to the snmp master agent.

While it is possible to implement non-local sub-agents, administrators should understand the security risks that are involved with this configuration. Primarily, because there is no communication authentication or privacy between the UnboundID® Directory Proxy Server and the master agent. An eavesdropper might be able to listen in on the monitoring data sent by the UnboundID® Directory Proxy Server. Likewise, a rogue sub-agent might be able to connect to the master agent and provide false monitoring data or deny access to SNMP monitoring data.

In general, it is recommended that sub-agents be located on the same host as the master agent.

MIBS

The Directory Proxy Server provides SMIv2-compliant MIB definitions (RFC 2578, 2579, 2580) for distinct monitoring statistics. These MIB definitions are to be found in text files under `resource/mib` directory under the server root directory.

Each MIB provides managed object tables for each specific SNMP management information as follows:

- **LDAP Remote Server MIB**. Provides information related to the health and status of the LDAP servers that the proxy server connects to, and statistics about the operations invoked by the proxy server on those LDAP servers.
- **LDAP Statistics MIB**. Provides a collection of connection-oriented performance data that is based on a connection handler in the Directory Proxy Server. A server typically contain only one connection handler and therefore supplies only one table entry.
- **Local DB Backend MIB**. Provides key metrics related to the state of the local database backends contained in the server.
- **Processing Time MIB**. Provides a collection of key performance data related to the processing time of operations broken down by several criteria but reported as a single aggregated data set.
- **Replication MIB**. Provides key metrics related to the current state of replication, which can help diagnose how much outstanding work replication may have to do.
- **System Status MIB**. Provides a set of critical metrics for determining the status and health of the system in relation to its work load.

For information on the available monitoring statistics for each MIB available on the directory server and the directory proxy server, see the text files provided in the `resource/mib` directory below the server installation.

The Directory Proxy Server generates an extensive set of SNMP traps for event monitoring. The traps display the severity, description, name, OID, and summary. For information about the available alert types for event monitoring, see the `resource/mib/UNBOUNDID-ALERT-MIB.txt` file.

Profiling Server Performance Using the Periodic Stats Logger

The Directory Proxy Server ships with a built-in Periodic Stats Logger that is useful for profiling server performance for a given configuration. At a specified interval, the Periodic Stats logger writes server statistics to a log file in a comma-separated format (.csv), which can be read by spreadsheet applications. The logger has a negligible impact on server performance unless the `log-interval` property is set to a very small value (less than 1 second).

You can control what statistics are logged and their verbosity. We recommend that you see what stats are available by experimenting with configurations or values. For example, you can make a configuration change in a test environment and see what stats are logged in the file.

To Enable the Periodic Stats Logger

By default, the Directory Server ships with the built-in 'Periodic Stats Logger' disabled. To enable it using the `dsconfig` tool or the web console, go to **Plugins** menu (available on the Advanced object menu), and then, select Stats Logger.

1. Run `dsconfig` in interactive mode. Enter the LDAP or LDAPS connection parameters when prompted.

   ```
   $ bin/dsconfig
   ```

2. Enter `o` to change to the Advanced Objects menu.

3. On the **Directory Server configuration console** menu, enter the number for Plugins.

4. On the **Plugin management** menu, enter the number corresponding to view and edit an existing plug-in.

5. On the **Plugin selection** list, enter the number corresponding to the Stats Logger.

6. On the **Periodic Stats Logger Plugin** menu, enter the number to set the `enabled` property to `TRUE`. When done, enter `f` to save and apply the configuration. The default logger will log information about the server every second to `<server-root>/logs/dsstats.csv`. If the server is idle, nothing will be logged, but this can be changed by setting the `suppress-if-idle` property to `FALSE` (`suppress-if-idle=false`).

   ```
   >>>> Configure the properties of the Periodic Stats Logger Plugin

          Property                 Value(s)
          -------------------------------------------------------------------------
     1)   description              Logs performance stats to a log file periodically.
     2)   enabled                  true
     3)   log-interval             1 s
     4)   collection-interval      200 ms
     5)   suppress-if-idle         true
     6)   header-prefix-per-column  false
     7)   empty-instead-of-zero     true
     8)   lines-between-header      50
     9)   included-ldap-stat        active-operations, num-connections,
                                      op-count-and-latency, work-queue
   ```

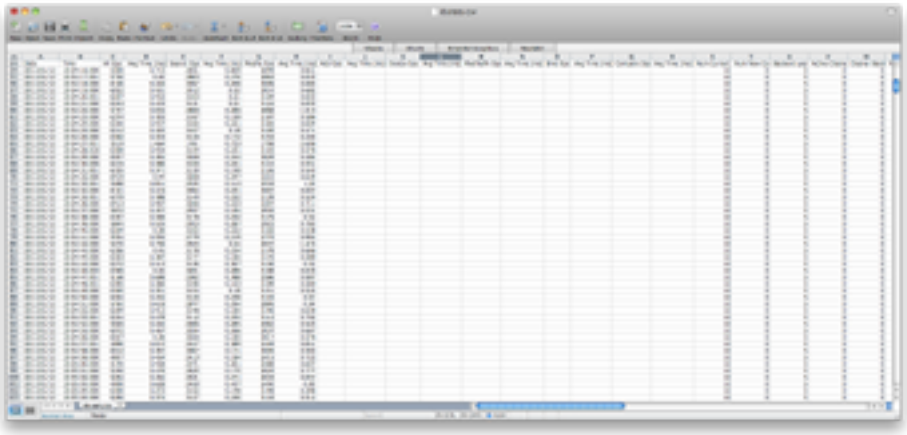
```
    10)  included-resource-stat    memory-utilization
    11)  histogram-format          count
    12)  histogram-op-type         all
    13)  local-db-backend-info     basic
    14)  replication-info          basic
    15)  entry-cache-info          -
    16)  log-file                  logs/dsstats.csv
    17)  log-file-permissions      640
    18)  append                    true
    19)  rotation-policy           Fixed Time Rotation Policy, Size Limit Rotation
                                     Policy
    20)  retention-policy          File Count Retention Policy

    ?)   help
    f)   finish - apply any changes to the Periodic Stats Logger Plugin
    a)   hide advanced properties of the Periodic Stats Logger Plugin
    d)   display the equivalent dsconfig command lines to either re-create this
           object or only to apply pending changes
    b)   back
    q)   quit

Enter choice [b]:
```

7. Run the Directory Proxy Server. For example, if you are running in a test environment, you can run the `search-and-mod-rate` tool to apply some searches and modifications to the server. You can run `search-and-mod-rate --help` to see an example command.

8. View the Stats log output at `<server-root>/logs/dsstats.csv`. You can open the file in a spreadsheet. The following image displays a portion of the file's output. On the actual file, you will need to scroll right for more statistics.



To Configure Multiple Periodic Stats Loggers

Multiple Periodic Stats Loggers can be created to log different statistics or to create a log at different intervals (such as logging cumulative operations statistics every hour). To create a new log, use the existing Stats Logger as a template to get reasonable settings, including rotation and retention policy.

1. Run `dsconfig` by repeating steps 1–3 in *To Enable the Periodic Stats Logger*.

2. From the **Plugin management** menu, enter the number to create a new plug-in.

3. From the **How to Create a New Plugin** menu, enter `t` to use an existing plug-in as a template.

4. Enter the number corresponding to the existing stats logger as a template.

5. Next, enter a descriptive name for the new stats logger. For this example, type `Stats Logger-10s`.

6. Enter the log file path to the file. For this example, type `logs/dsstats2.csv`.

7. On the **Periodic Stats Logger Plugin** menu, make any other change to your logger. For this example, change the `log-interval` to 10s, and the `suppress-if-idle` to false. When finished, enter `f` to save and apply the configuration.

8. You should now see two loggers `dsstats.csv` and `dsstats2.csv` in the `logs` directory.

Adding Custom Logged Statistics to the Periodic Stats Logger

You can add custom statistics based on any attribute in any entry under `cn=monitor` using the Custom Logged Stats object. This configuration object provides powerful controls for how monitor attributes are written to the log. For example, you can extract a value from a monitor attribute using a regular expression. Newly created Custom Logged Stats will automatically be included in the Periodic Stats Logger output.

Besides allowing a straight pass-through of the values using the 'raw' statistic-type, you can configure attributes to be treated as a counter (where the interval includes the difference in the value since the last interval), an average, a minimum, or a maximum value held by the attribute during the specified interval. The value of an attribute can also be scaled by a fixed value or by the value of another monitor attribute.

Note: Custom third-party server extensions that were written using the Server SDK can also expose interval statistics using the Periodic Stats Logger. The extension must first implement the SDK's `MonitorProvider` interface and register with the server. The monitor attributes produced by this custom `MonitorProvider` are then available to be referenced by a Custom Logged Stats object.

To illustrate how to configure a Custom Logged Statistics Logger, the following procedure reproduces the built-in "Consumer Total GB" column that shows up in the output when the `included-resource-stat` property is set to memory-utilization on the Periodic Stats Logger. The column is derived from the `total-bytes-used-by-memory-consumers` attribute of the `cn=JVM Memory Usage,cn=monitor` entry as follows:

```
dn: cn=JVM Memory Usage,cn=monitor
objectClass: top
objectClass: ds-monitor-entry
objectClass: ds-memory-usage-monitor-entry
objectClass: extensibleObject
cn: JVM Memory Usage
...
total-bytes-used-by-memory-consumers: 3250017037
```

To Configure a Custom Logged Statistic Using dsconfig Interactive

1. Run `dsconfig` and enter the LDAP/LDAPS connection parameters when prompted.

```
$ bin/dsconfig
```

2. On the Directory Server configuration main menu (Advanced Objects menu), enter the number corresponding to Custom Logged Stats.

3. On the Custom Logged Stats menu, enter the number corresponding to Create a new Custom Logged Stats.

4. Select the Periodic Stats Logger Plugin from the list if more than one is present on the system. If you only have one stats logger, press **Enter** to confirm that you want to use the existing plugin.

5. Enter a descriptive name for the Custom Logged Stats. For this example, enter `Memory Usage`.

6. From the `monitor-objectclass` property menu, enter the objectclass attribute to monitor. For this example, enter `ds-memory-usage-monitor-entry`. You can run `ldapsearch` using the base DN "cn=JVM Memory Usage,cn=monitor" entry to view the entry.

7. Next, specify the attributes of the monitor entry that you want to log in the stats logger. In this example, enter `total-bytes-used-by-memory-consumers`, and then press **Enter** again to continue.

8. Next, specify the type of statistics for the monitored attribute that will appear in the log file. In this example, enter the option for raw statistics as recorded by the logger.

9. In the Custom Logged Stats menu, review the configuration. At this point, we want to set up a column name that lists the Memory Usage. Enter the option to change the `column-name` property.

10. Next, we want to add a specific label for the column name. Enter the option to add a value, and then enter **Memory Consumer Total (GB)**, and press **Enter** again to continue.

11. Confirm that you want to use the `column-name` value that you entered in the previous step, and then press `Enter` to use the value.

12. Next, we want to scale the Memory Consumer Totals by one gigabyte. On the **Custom Logged Stats** menu, enter the option to change the `divide-value-by` property.

13. On the `divide-value-by` property menu, enter the option to change the value, and then enter 1073741824 (i.e., 1073741824 bytes = 1 gigabytes).

14. On the **Custom Logged Stats** menu, review your configuration. When finished, enter `f` to save and apply the settings.

```
>>>> Configure the properties of the Custom Logged Stats
 >>>> via creating 'Memory Usage' Custom Logged Stats
```

```
         Property                   Value(s)
         -----------------------------------------------------------
    1)   description                -
    2)   enabled                    true
    3)   monitor-objectclass        ds-memory-usage-monitor-entry
    4)   include-filter             -
    5)   attribute-to-log           total-bytes-used-by-memory-consumers
    6)   column-name                Memory Consumer Total (GB)
    7)   statistic-type             raw
    8)   header-prefix              -
    9)   header-prefix-attribute    -
    10)  regex-pattern              -
    11)  regex-replacement          -
    12)  divide-value-by            1073741824
    13)  divide-value-by-attribute  -
    14)  decimal-format             #.##
    15)  non-zero-implies-not-idle  false

    ?)   help
    f)   finish - create the new Custom Logged Stats
    a)   hide advanced properties of the Custom Logged Stats
    d)   display the equivalent dsconfig arguments to create this object
    b)   back
    q)   quit

Enter choice [b]:
```

```
The Custom Logged Stats was created successfully
```

When the Custom Logged Stats configuration change is completed, the new stats value should immediately show up in the Periodic Stats Logger output file.

To Configure a Custom Periodic Stats Logger Using dsconfig Non-Interactive

- Use the `dsconfig` non-interactive command-line equivalent to create your custom stats logger. The following one-line command replicates the procedure in the previous section. This command produces a column named "Memory Consumer Total (GB)" that contains the value of the of `total-bytes-used-by-memory-consumers` attribute pulled from the entry with the `ds-memory-usage-monitor-entry` objectclass. This value is scaled by 1073741824 to get to a value represented in GBs.

```
$ bin/dsconfig create-custom-logged-stats --plugin-name "Stats Logger" \
  --stats-name "Memory Usage" --type custom \
  --set monitor-objectclass:ds-memory-usage-monitor-entry \
  --set attribute-to-log:total-bytes-used-by-memory-consumers \
  --set "column-name:Memory Consumer Total (GB)" --set statistic-type:raw \
  --set divide-value-by:1073741824
```

Working with Administrative Alert Handlers

The UnboundID® Directory Proxy Server provides mechanisms to send alert notifications to administrators when significant problems or events occur during processing, such as problems during server startup or shutdown. The Directory Proxy Server provides a number of alert handler implementations, including:

- **Error Log Alert Handler**. Sends administrative alerts to the configured server error logger(s).
- **Exec Alert Handler**. Executes a specified command on the local system if an administrative alert matching the criteria for this alert handler is generated by the Directory Proxy Server.

Information about the administrative alert will be made available to the executed application as arguments provided by the command.

- **Groovy Scripted Alert Handler**. Provides alert handler implementations defined in a dynamically-loaded Groovy script that implements the `ScriptedAlertHandler` class defined in the Server SDK.
- **JMX Alert Handler**. Sends administrative alerts to clients using the Java Management Extensions (JMX) protocol. UnboundID uses JMX for monitoring entries and requires that the JMX connection handler be enabled.
- **SMTP Alert Handler**. Sends administrative alerts to clients via email using the Simple Mail Transfer Protocol (SMTP). The server requires that one or more SMTP servers be defined in the global configuration.
- **SNMP Alert Handler**. Sends administrative alerts to clients using the Simple Network Monitoring Protocol (SNMP). The server must have an SNMP agent capable of communicating via SNMP 2c.
- **SNMP Subagent Alert Handler**. Sends SNMP traps to a master agent in response to administrative alerts generated within the server.
- **Third Party Alert Handler**. Provides alert handler implementations created in third-party code using the Server SDK.

Configuring the JMX Connection Handler and Alert Handler

You can configure the JMX connection handler and alert handler respectively using the `dsconfig` tool. Any user allowed to receive JMX notifications must have the `jmx-read` and `jmx-notify` privileges. By default, these privileges are not granted to any users (including root users or global administrators). For security reasons, we recommend that you create a separate user account that does not have any other privileges but these. Although not shown in this section, you can configure the JMX connection handler and alert handler using `dsconfig` in interactive command-line mode, which is visible on the "Standard" object menu.

To Configure the JMX Connection Handler

1. Use `dsconfig` to enable the JMX Connection Handler.

```
$ bin/dsconfig set-connection-handler-prop \
  --handler-name "JMX Connection Handler" \
  --set enabled:true \
  --set listen-port:1689
```

2. Add a new non-root user account with the `jmx-read` and `jmx-notify` privileges. This account can be added using the `ldapmodify` tool using an LDIF representation like:

```
dn: cn=JMX User,cn=Root DNs,cn=config
changetype: add
objectClass: top
objectClass: person
objectClass: organizationalPerson
objectClass: inetOrgPerson
objectClass: ds-cfg-root-dn-user
givenName: JMX
```

```
sn: User
cn: JMX User
userPassword: password
ds-cfg-inherit-default-root-privileges: false
ds-cfg-alternate-bind-dn: cn=JMX User
ds-privilege-name: jmx-read
ds-privilege-name: jmx-notify
```

To Configure the JMX Alert Handler

- Use `dsconfig` to configure the JMX Alert Handler.

```
$ bin/dsconfig set-alert-handler-prop --handler-name "JMX Alert Handler" \
  --set enabled:true
```

Configuring the SMTP Alert Handler

By default, there is no configuration entry for an SMTP alert handler. To create a new instance of an SMTP alert handler, use the `dsconfig` tool.

Configuring the SMTP Alert Handler

- Use the `dsconfig` tool to configure the SMTP Alert Handler.

```
$ bin/dsconfig create-alert-handler \
  --handler-name "SMTP Alert Handler" \
  --type smtp \
  --set enabled:true \
  --set "sender-address:alerts@example.com" \
  --set "recipient-address:administrators@example.com" \
  --set "message-subject:Directory Admin Alert \%\%alert-type\%\%" \
  --set "message-body:Administrative alert:\\n\%\%alert-message\%\%"
```

Configuring the SNMP Subagent Alert Handler

You can configure the SNMP Subagent alert handler using the `dsconfig` tool, which is visible at the "Standard" object menu. Before you begin, you need an SNMP Subagent capable of communicating via SNMP2c. For more information on SNMP, see Monitoring Using SNMP.

To Configure the SNMP Subagent Alert Handler

- Use `dsconfig` to configure the SNMP subagent alert handler. The `server-host-name` is the address of the system running the SNMP subagent. The `server-port` is the port number on which the subagent is running. The `community-name` is the name of the SNMP community that is used for the traps.

  The Directory Proxy Server also supports a SNMP Alert Handler, which is used in deployments that do not enable an SNMP subagent.

```
$ bin/dsconfig set-alert-handler-prop \
  --handler-name "SNMP Subagent Alert Handler" \
  --set enabled:true \
  --set server-host-name:host2 \
  --set server-port:162 \
```

```
  --set community-name:public
```

Working with Virtual Attributes

The UnboundID® Directory Proxy Server provides dynamically generated attributes called virtual attributes for local proxy server data. The proxy virtual attributes apply to a local proxy backend, such as `cn=config` or the Root DSE. If you want to have virtual attributes in entries for proxied requests, then they must be configured in the backend servers. Alternately, attributes may be inserted into those entries using proxy transformations. For more information about configuring proxy transformations, see “Configuring Proxy Transformations”.

For example, you can define a virtual attribute and assign it to the Root DSE as follows:

```
$ bin/dsconfig create-virtual-attribute \
  --name defineDescriptionOnRootDSE --type user-defined \
  --set enabled:true --set attribute-type:description \
  --set filter:objectclass=ds-root-dse --set value:PrimaryProxy
```

If you search the Root DSE using the following LDAP search, you see that the description attribute now has the value `PrimaryProxy`.

```
$ bin/ldapsearch --baseDN "" --searchScope base --bindDN "" \
  --bindPassword "" --port 5389 -- hostname localhost \
  "objectclass=*" description

dn:
description:PrimaryProxy
```

Managing Directory Proxy Server Extensions

You can create extensions that use the Server SDK to extend the functionality of your Directory Proxy Server. Extension bundles are installed from a .zip archive or a file system directory. You can use the `manage-extension` tool to install or update any extension that is packaged using the extension bundle format. It opens and loads the extension bundle, confirms the correct extension to install, stops the server if necessary, copies the bundle to the server install root, and then restarts the server.

Note: The `manage-extension` tool may only be used with Java extensions packaged using the extension bundle format. Groovy extensions do not use the extension bundle format. For more information, see the "Building and Deploying Java-Based Extensions" section of the Server SDK documentation, which describes the extension bundle format and how to build an extension.

Chapter 5 Managing Access Control

The UnboundID® Directory Proxy Server provides a fine-grained access control model to ensure that users are able to access the information they need, but are prevented from accessing information that they should not be allowed to see. It also includes a privilege subsystem that provides even greater flexibility and protection in many key areas.

This chapter presents the access control model and how it applies to the directory proxy server.

Topics:

Overview of Access Control

Access controls enforce access wherever data is stored. For accesses through the proxy server to backend directory server data, access control is enforced on the directory server level. If you configure global access controls in the directory proxy server for data stored on backend directory servers, they will not be enforced. However, you can configure global access controls in the directory proxy server to control data stored on the directory proxy servers, including schema, configuration, monitoring and alert information, and administrative data. You can configure what subtree view clients can see and the operations they can perform at a coarse level using client connection policies. For more information about client connection policies, see *Configuring Client Connection Policies*.

The access control model uses access control instructions (ACIs), which are stored in the `aci` operational attribute, to determine what a user or a group of users can do with a set of entries, down to the attribute level. The operational attribute can appear on any entry and will affect the entry or any subentries within that branch of the directory information tree (DIT).

Configuring Access Control with Entry Balancing

The directory servers in each backend server set should contain a copy of all the access control rules applicable to the branch of the tree under the entry balancing point, or the entry balancing base entry. These access controls should be stored in ACI attributes at or above the entry balancing point. The Directory Proxy Server ensures that any changes to entries within the scope of the entry balancing request processor, but outside the balancing point, are applied to all backend server sets. This means, for example, that any ACI stored at the entry balancing point will be kept in sync if changes are made through the Directory Proxy Server.

Entry balancing introduces a possible issue when clients to the Directory Proxy Server authenticate as users whose entries are among those balanced. The access control rules may require access to the entry contents of the user issuing the request. By its nature, entry balancing dictates that servers in only one of the backend sets contain the entry for a given user. If the server processing a request does not contain the issuing user's entry, then the access control cannot be evaluated. As a solution to this problem, the Directory Proxy Server allows an alternate authorization identity to be defined for any user, where the alternate identity is the DN of an entry that exists in all directory servers in all backend sets. The alternate authorization identity is used when the directory proxy server observes that the directory server processing a request does not contain the entry of the user issuing the request.

To use alternate authorization identities, specify the name of an attribute in user entries to hold the alternate DN. This attribute is specified by the `authz-attribute` property of the entry balancing request processor configuration. This advanced property has a default value of `ds-authz-map-to-dn`, an attribute reserved for this purpose.

Next, define one or more generic users to be used as alternate authorization identities. For example, suppose the entry balancing point is the `ou=people` entry under `dc=example,dc=com`. You could create the following users:

```
uid=normal-user,dc=example,dc=com
uid=server-admin,dc=example,dc=com
uid=password-admin,dc=example,dc=com
```

Note that there must be a copy of each generic user in all of the backend directory servers. Make sure to give these generic users the set of rights that should be granted to those different classes of user.

Finally, every user among the balanced entries should be assigned one of these generic users as their alternate authorization identity. The directory server schema must allow the chosen attribute for alternate authorization to be present in user entries. For example, you could create an auxiliary object class containing `ds-authz-map-to-dn` as an allowed attribute, and add this object class value to all user entries. Then, you could explicitly add the following attribute value to a server-admin user:

```
ds-authz-map-to-dn: uid=server-admin,dc=example,dc=com
```

Alternatively, the attribute could be supplied as a virtual attribute definition. If any user among the balanced entries does not have an alternate authorization identity defined, the proxy server will instead use the value of the `authz-dn` property of the entry balancing request processor configuration, if provided. For example, the following command could be used to specify that users are “normal” users by default (and thus avoid having to explicitly assign each of them an alternate identity).

```
$ 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"
```

Suppose you have set things up such that `uid=user.0,ou=people,dc=example,dc=com` is a normal user. If an operation performed by this user is routed to a directory server containing this user entry, then you will see `authzDN="uid=user.0,ou=People,dc=example,dc=com"` in the access log for the operation in that directory server. The sample output has been wrapped for readability.

```
% bin/ldapsearch -D "uid=user.0,ou=people,dc=example,dc=com" \
  -w password -b uid=user.0,ou=people,dc=example,dc=com "(objectclass=*)"

[15/Jun/2009:11:51:23  -0500] BIND REQUEST conn=171 op=1 msgID=2 version="3"
dn="uid=user.0,ou=people,dc=example,dc=com" authType="SIMPLE"
[15/Jun/2009:11:51:23 -0500] BIND RESULT conn=171 op=1 msgID=2 resultCode=0 etime=1.826
authDN="uid=user.0,ou=People, dc=example,dc=com"
[15/Jun/2009:11:51:23 -0500] SEARCH REQUEST conn=172 op=1 msgID=2
via="app='UnboundID-Proxy' address='127.0.0.1'
authzID='dn:uid=user.0,ou=people,dc=example,dc=com' sessionID='conn=1' requestID='op=1'"
base="uid=user.0, ou=people,dc=example,dc=com" scope=2 filter="(objectclass=*)"
 attrs="ALL"
[15/Jun/2009:11:51:23 -0500] SEARCH RESULT conn=172 op=1 msgID=2 resultCode=0
 etime=3.101
entriesReturned=1 authzDN="uid=user.0,ou=People,dc=example,dc=com"
```

However, if an operation performed while bound as that same user is routed to a directory server that does not contain the user entry, then you will see `authzID='dn:uid=normal-user,dc=example,dc=com'` in the directory server log, indicating that the alternate authorization identity was used. In this example, it is assumed that user.1 is in a different backend set from user.0:

```
$ bin/ldapsearch -D "uid=user.0,ou=people,dc=example,dc=com" -w password -b
  uid=user.1,ou=people,dc=example,dc=com "(objectclass=*)"

[15/Jun/2009:11:54:35 -0500] SEARCH REQUEST conn=153 op=1 msgID=2
via="app='UnboundID-Proxy' address='127.0.0.1'
authzID='dn:uid=normal-user,dc=example,dc=com' sessionID='conn=2' requestID='op=1'"
```

```
base="uid=user.1,ou=people,dc=example,dc=com" scope=2 filter="(objectclass=*)"
 attrs="ALL"
[15/Jun/2009:11:54:35 -0500] SEARCH RESULT conn=153 op=1 msgID=2 resultCode=0
 etime=2.038
entriesReturned=1 authzDN="uid=normal-user,dc=example,dc=com"
```

In the above output, you do not see the bind operation in the log because that operation was performed on the other server.

Overview of Access Control

Access control determines what a user or a group of users can do with a set of entries, down to the attribute level. The Directory Proxy Server uses access control instructions (ACIs) to allow or deny privileges users access to the entry or subentries and stores them 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 specifies four items:

- **Resources**. *Resources* are the targeted items or objects that specifies the set of entries and/or operations to which the access control instruction applies. For example, you can specify access to certain attributes, such as the `cn` or `userPassword` password.
- **Name**. *Name* is the descriptive label for each access control instruction. Typically, you will have multiple access control instructions for a given branch of your DIT. The access control name helps describe its purpose. For example, you can configure an access control instructions labelled "ACI to grant full access to administrators."
- **Clients**. *Clients* are the users or entities to which you grant or deny access. You can specify individual users or groups of users using an LDAP URL. For example, you can specify a group of administrators using the LDAP URL: `groupdn="ldap:///cn=admins,ou=groups,dc=example,dc=com."`
- **Rights**. *Rights* are permissions granted to users or client applications. You can grant or deny access to certain branches or operations. For example, you can grant `read` or `write` permission to a `telephoneNumber` attribute.

Key Access Control Features

The UnboundID® Directory Proxy Server provides important access control features that provide added security for the directory server's entries.

Improved Validation and Security

The Directory Proxy Server provides an access control model with strong validation to help ensure that invalid ACIs are not allowed into the server. For example, the Directory Proxy Server ensures that all access control rules added over LDAP are valid and can be fully parsed. Any operation that attempts to store one or more invalid ACIs are rejected. The same validation is applied to ACIs contained in data imported from an LDIF file. Any entry containing a malformed `aci` value will be rejected.

As an additional level of security, the Directory Proxy Server examines and validates all ACIs stored in the data whenever a backend is brought online. If any malformed ACIs are found in the backend, then the server generates an administrative alert to notify administrators of the problem and places itself in lockdown mode. While in lockdown mode, the server only allows requests from users who have the `lockdown-mode` privilege. This action allows administrators to correct the malformed ACI while ensuring that no sensitive data is inadvertently exposed due to an access control instruction not being enforced. When the problem has been corrected, the administrator can use the `leave-lockdown-mode` tool or restart the server to allow it to resume normal operation.

Global ACIs

Global ACIs are a set of ACIs that can apply to entries anywhere in the directory (although they can also be scoped so that they only apply to a specific set of entries). They work in conjunction with access control rules stored in user data and provide a convenient way to define ACIs that span disparate portions of the DIT.

In the UnboundID® Directory Proxy Server, global ACIs are defined within the server configuration, in the `global-aci` property of configuration object for the access control handler. They can be viewed and managed using configuration tools like `dsconfig` and the web administration console.

The global ACIs available by default in the UnboundID® Directory Proxy Server include:

- Allow anyone (including unauthenticated users) to access key attributes of the root DSE, including: `namingContexts`, `subschemaSubentry`, `supportedAuthPasswordSchemes`, `supportedControl`, `supportedExtension`, `supportedFeatures`, `supportedLDAPVersion`, `supportedSASLMechanisms`, `vendorName`, and `vendorVersion`.
- Allow anyone (including unauthenticated users) to access key attributes of the subschema subentry, including: `attributeTypes`, `dITContentRules`, `dITStructureRules`, `ldapSyntaxes`, `matchingRules`, `matchingRuleUse`, `nameForms`, and `objectClasses`.
- Allow anyone (including unauthenticated users) to include the following controls in requests made to the server: authorization identity request, manage DSA IT, password policy, real attributes only, and virtual attributes only.
- Allow anyone (including unauthenticated users) to request the following extended operations: get symmetric key, password modify request, password policy state, StartTLS, and Who Am I?

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 directory subtree, such as `dc=example,dc=com`, and applies to that base entry and all entries below it in the tree. The Directory Proxy Server iterates through the DIT to compile the access control rules into an internally-used list of denied and allowed targets and their permissable operations. When a client application, such as `ldapsearch`, enters a request, the Directory Proxy Server checks that the user who binds with the server has the necessary access rights to the requested search targets. ACIs are cumulatively applied, so that a user who may have an ACI at an entry, may

also have other access rights available if ACIs are defined higher in the DIT and are applicable to the user. In most environments, ACIs are defined at the root of a main branch or a subtree, and not on individual entries unless absolutely required.

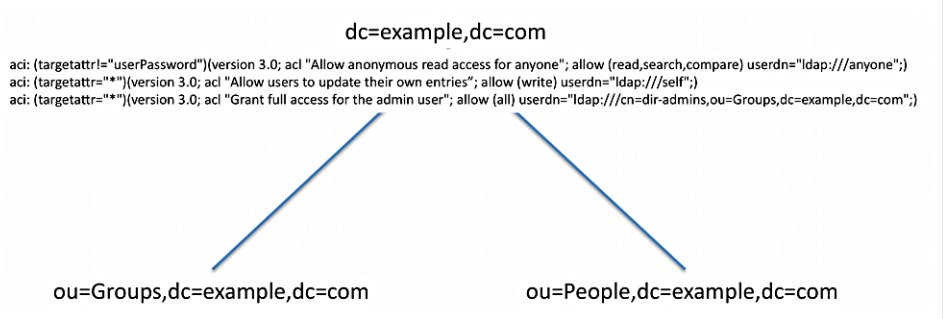


Figure 5: ACI

An access control rule has a basic syntax as follows:

```
aci : (targets) (version 3.0; acl "name"; permissions bind rules;)
```

Table 3: Access Control Components

Access Control Component	Description
targets	Specifies the set of entries and/or attributes to which an access control rule applies. Syntax: *(target keyword = || != expression)*
name	Specifies the name of the ACI.
permissions	Specifies the type of operations to which an access control rule might apply. Syntax: *allow||deny (permission)*
bind rules	Specifies the criteria that indicate whether an access control rule should apply to a given requestor. Syntax: *bind rule keyword = ||!= expression;*. The bind rule syntax requires that it be terminated with a ";".

Summary of Access Control Keywords

This section provides an overview of the keywords supported for use in the UnboundID® Directory Proxy Server access control implementation.

Targets

A target expression specifies the set of entries and/or attributes to which an access control rule applies. The *keyword* specifies the type of target element. The *expression* specifies the items that is targeted by the access control rule. The operator is either the equal ("=") or not-equal ("!="). Note that the "!=" operator cannot be used with `targattrfilters` and `targetscope` keywords. For specific examples on each target keyword, see the section *Working with Targets*.

```
(keyword [=||!=]expression)
```

The following keywords are supported for use in the target portion of ACIs:

Table 4: Summary of Access Control Target Keywords

Target Keyword	Description	Wildcards
extop	Specifies the OIDs for any extended operations to which the access control rule should apply.	No
target	Specifies the set of entries, identified using LDAP URLs, to which the access control rule applies.	**Yes**
targattrfilters	Identifies specific attribute values based on filters that may be added to or removed from entries to which the access control rule applies.	**Yes**
targetattr	Specifies the set of attributes to which the access control rule should apply.	**Yes**
targetcontrol	Specifies the OIDs for any request controls to which the access control rule should apply.	No
targetfilter	Specifies one or more search filters that may be used to indicate the set of entries to which the access control should apply.	**Yes**
targetscope	Specifies the scope of entries, relative to the defined target entries or the entry containing the ACI fi there is no target, to which the access control rule should apply.	No

Permissions

Permissions indicate the types of operations to which an access control rule might apply. You can specify if the user or group of users are allowed or not allowed to carry out a specific operation. For example, you would grant read access to a targeted entry or entries using "allow (read)" permission. Or you can specifically deny access to the target entries and/or attributes using the "deny (read)" permission. You can list out multiple permissions as required in the ACI.

```
allow (permission1 ...,permission2,...permissionN)
```

```
deny (permission1 ...,permission2,...permissionN)
```

The following keywords are supported for use in the permissions portion of ACIs:

Table 5: Summary of Access Control Permissions

Permission	Description
add	Indicates that the access control should apply to add operations.
compare	Indicates that the access control should apply to compare operations, as well as to search operations with a base-level scope that targets a single entry.
delete	Indicates that the access control should apply to delete operations.
export	Indicates that the access control should only apply to modify DN operations in which an entry is moved below a different parent by specifying a new superior DN in the modify DN request. The requestor must have the `export` permission for operations against the entry's original DN. The requestor must have the `import` permission for operations against the entry's new superior DN. For modify DN operations that merely alter the RDN of an entry but keeps it below the same parent (i.e., renames the entry), only the `write` permission is required. This is true regardless of whether the entry being renamed is a leaf entry or has subordinate entries.
import	See the description for the `export` permission.
proxy	Indicates that the access control rule should apply to operations that attempt to use an alternate authorization identity (for example, operations that include a proxied authorization request control, an intermediate client request control with an alternate authorization identity, or a client that

Permission	Description
	has authenticated with a SASL mechanism that allows an alternate authorization identify to be specified).
read	Indicates that the access control rule should apply to search result entries returned by the server.
search	Indicates that the access control rule should apply to search operations with a non-base scope.
selfwrite	Indicates that the access control rule should apply to operations in which a user attempts to add or remove his or her own DN to the values for an attribute (for example, whether users may add or remove themselves from groups).
write	Indicates that the access control rule should apply to modify and modify DN operations.
all	An aggregate permission that includes all other permissions except “proxy.” This is equivalent to providing a permission of “add, compare, delete, read, search, selfwrite, write.”

Bind Rules

The Bind Rules indicate whether an access control rule should apply to a given requester. The syntax for the target keyword is shown below. The *keyword* specifies the type of target element. The expression specifies the items that is targeted by the access control rule. The operator is either equals ("=") or not-equals ("!="). The semi-colon delimiter symbol (";") is required after the end of the final bind rule.

```
keyword [=||!= ] expression;
```

Multiple bind rules can be combined using boolean operations (AND, OR, NOT) for more access control precision. The standard Boolean rules for evaluation apply: innermost to outer parentheses first, left to right expressions, NOT before AND or OR. For example, an ACI that includes the following bind rule targets all users who are not `uid=admin,dc=example,dc=com` and use simple authentication.

```
(userdn!="ldap:///uid=admin,dc=example,dc=com" and authmethod="simple");
```

The following bind rule targets the `uid=admin,dc=example,dc=com` and authenticates using SASL EXTERNAL or accesses the server from a loopback interface.

```
(userdn="ldap:///uid=admin,dc=example,dc=com and (authmethod="SSL" or ip="127.0.0.1"));
```

The following keywords are supported for use in the bind rule portion of ACIs:

Table 6: Summary of Bind Rule Keywords

Bind Rule Keyword	Description
authmethod	Indicates that the requester’s authentication method should be taken into account when determining whether the access control rule should apply to an operation. Wildcards are not allowed in this expression. The keyword’s syntax is as follows: **`authmethod`** `=` *`method`* where *method* is one of the following representations: ➢ none ➢ simple. Indicates that the client is authenticated to the server using a bind DN and password. ➢ ssl. Indicates that the client is authenticated with an SSL/TLS certificate (e.g., via SASL EXTERNAL), and not just over a secure connection to the server. ➢ sasl {sasl_mechanism}. Indicates that the client is authenticated to the server using a specified SASL Mechanism.

Bind Rule Keyword	Description

	The following example allows users who authenticate with an SSL/TLS certificate (e.g., via SASL EXTERNAL) to update their own entries:
	
	`aci: (targetattr="*")`
	`  (version 3.0; acl "Allow users to update their own entries";`
	`   allow (write) (userdn="ldap:///self" and authmethod="ssl");)`

dayofweek	Indicates that the day of the week should be taken into account when determining whether the access control rule should apply to an operation. Wildcards are not allowed in this expression. Multiple day of week values may be separated by commas. The keyword's syntax is as follows:
	
	dayofweek = *day1, day2, ...*
	
	where *day* is one of the following representations:
	
	➢ sun
	➢ mon
	➢ tues
	➢ wed
	➢ thu
	➢ fri
	➢ sat
	
	The following example allows users who authenticate with an SSL/TLS certificate (e.g., via SASL EXTERNAL) on weekdays to update their own entries:
	
	`aci: (targetattr="*")`
	`  (version 3.0; acl "Allow users to update their own entries";`
	`   allow (write) (dayofweek!="sun,sat" and userdn="ldap:///self"`
	`   and authmethod="ssl");)`

dns	Indicates that the requester's DNS-resolvable host name should be taken into account when determining whether the access control rule should apply to an operation. Wildcards are allowed in this expression. Multiple DNS patterns may be separated by commas. The keyword's syntax is as follows:
	
	dns = *dns-host-name*
	
	The following example allows users on hostname `server.example.com` to update their own entries:
	
	`aci: (targetattr="*")`
	`  (version 3.0; acl "Allow users to update their own entries";`
	`   allow (write) (dns="server.example.com" and userdn="ldap:///self");)`

groupdn	Indicates that the requester's group membership should be taken into account when determining whether the access control rule should apply to any operation. Wildcards are not allowed in this expression.
	
	groupdn [= || !=] "ldap:///groupdn [|| ldap:///groupdn] ..."
	
	The following example allows users in the managers group to update their own entries:
	
	`aci: (targetattr="*")`
	`  (version 3.0; acl "Allow users to update their own entries";`
	`   allow (write)`
	`   (groupdn="ldap:///cn=managers,ou=groups,dc=example,dc=com");)`

ip	Indicates that the requester's IP address should be taken into account when determining whether the access control rule should apply to an operation. Wildcards are allowed in this expression. Multiple IP address patterns may be separated by commas. The keyword's syntax is as follows:

Bind Rule Keyword	Description

	ip [= || !=] *ipAddressList*
	where *ipAddressList* is one of the following representations:
	➢ A specific IPv4 address: 127.0.0.1
	➢ An IPv4 address with wildcards to specify a subnetwork: 127.0.0.*
	➢ An IPv4 address or subnetwork with subnetwork mask: 123.4.5.0+255.255.255.0
	➢ An IPv4 address range using CIDR notation: 123.4.5.0/24
	➢ An IPv6 address as defined by RFC 2373.
	The following example allows users on 10.130.10.2 and localhost to update their own entries:
	`aci: (targetattr="*")`
	`  (version 3.0; acl "Allow users to update their own entries";`
	`   allow (write) (ip="10.130.10.2,127.0.0.1" and userdn="ldap:///`
	`self");)`

timeofday	Indicates that the time of day should be taken into account when determining whether the access control rule should apply to an operation. Wildcards are not allowed in this expression. The keyword's syntax is as follows:
	`timeofday [ = || != || >= || > || <= || < ] time`
	where *time* is one of the following representations:
	➢ 4-digit 24-hour time format (0000 to 2359, where the first two digits represent the hour of the day and the last two represent the minute of the hour)
	➢ Wildcards are not allowed in this expression
	The following example allows users to update their own entries if the request is received before 12 noon.
	`aci: (targetattr="*")`
	`  (version 3.0; acl "Allow users who authenticate before noon`
	`    to update their own entries";`
	`    allow (write) (timeofday<1200 and userdn="ldap:///self"`
	`    and authmethod="simple");)`

userattr	Indicates that the requester's relation to the value of the specified attribute should be taken into account when determining whether the access control rule should apply to an operation. A bindType value of USERDN indicates that the target attribute should have a value which matches the DN of the authenticated user. A bindType value of GROUPDN indicates that the target attribute should have a value which matches the DN of a group in which the authenticated user is a member. A bindType value of LDAPURL indicates that the target attribute should have a value that is an LDAP URL whose criteria matches the entry for the authenticated user. Any value other than USERDN, GROUPDN, or LDAPURL is expected to be present in the target attribute of the authenticated user's entry. The keyword's syntax is as follows:
	`userattr = attrName# [ bindType || attrValue ]`
	where:
	➢ *attrName* = name of the attribute for matching
	➢ *bindType* = USERDN, GROUPDN, LDAPURL
	➢ *attrValue* = an attribute value
	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";`

Bind Rule Keyword	Description

	`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 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 entry at or below the specified ACI.
	
	`aci: (targetattr="*")`
	`(version 3.0; acl "Allow any member of Eng Dept to update any entry at or below the ACI";`
	`allow (write) userattr="department#ENGINEERING";)`
	
	The following example allows an entry to be updated by any user whose entry matches the criteria defined in the LDAP URL contained in the `allowedEditorCriteria` attribute of the target entry.
	
	`aci: (targetattr="*")`
	`(version 3.0; acl "Allow a user that matches the filter to change entries";`
	`allow (write) userattr="allowedEditorCriteria#LDAPURL";)`

userdn	Indicates that the user's DN should be taken into account when determining whether the access control rule should apply to an operation. The keyword's syntax is as follows:
	
	`userdn [ = || != ] "ldap:///value [ || "ldap:///value ..."]`
	
	where *value* is one of the following representations:
	
	➢ The DN of the target user
	➢ A value of `anyone` to match any client, including unauthenticated clients.
	➢ A value of `all` to match any authenticated client.
	➢ A value of `parent` to match the client authenticated as the user defined in the immediate parent of the target entry.
	➢ A value of `self` to match the client authenticated as the user defined in the target entry.
	
	If the value provided is a DN, then that DN may include wildcard characters to define patterns. A single asterisk will match any content within the associated DN component, and two consecutive asterisks may be used to match zero or more DN components.
	
	The following example allows users to update their own entries:
	
	`aci: (targetattr="*")`
	`(version 3.0; acl "Allow users to update their own entries";`
	`allow (write) userdn="ldap:///self";)`

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 UnboundID® Directory Proxy Server:

- **(targetattr="*")**. Indicates that the access control rule applies to all user attributes. Operational attributes will not automatically be included in this set.
- **(targetattr="+")**. Indicates that the access control rule applies to all operational attributes. User attributes will not automatically be included in this set.
- **(targetattr="attr1||attr2||attr3||...||attrN")**. Indicates that the access control rule applies only to the named set of attributes.
- **(targetattr!="attr1||attr2||attr3||...||attrN")**. Indicates that the access control rule applies to all user attributes except the named set of attributes. It will not apply to any operational attributes.

The targeted attributes can be classified as user attributes and operational attributes. User attributes define the actual data for that entry, while operational attributes provide additional metadata about the entry that can be used for informational purposes, such as when the entry was created, last modified and by whom. Metadata can also include attributes specifying which password policy applies to the user, or overrided default constraints like size limit, time limit, or look-through limit for that user.

The UnboundID® Directory Proxy Server distinguishes between these two types of attributes in its access control implementation. The Directory Proxy Server does not automatically grant any access at all to operational attributes. For example, the following clause applies only to user attributes and not to operational attributes:

```
(targetattr="*")
```

You can also target multiple attributes in the entry. The following clause targets the common name (cn), surname (sn) and state (st) attribute:

```
(targetattr="cn||sn||st")
```

You can use the "+" symbol to indicate that the rule should apply to all operational attributes, as follows:

```
(targetattr="+")
```

To include all user and all operational attributes, you use both symbols, as follows:

```
(targetattr="*||+")
```

If there is a need to target a specific operational attribute rather than all operational attributes, then it can be specifically included in the values of the `targetattr` clause, as follows:

```
(targetattr="ds-rlim-size-limit")
```

Or if you want to target all user attributes and a specific operational attribute, then you can use them in the `targetattr` clause, as follows:

```
(targetattr="*||ds-rlim-size-limit")
```

The following ACIs are placed on the `dc=example,dc=com` tree and allows any user anonymous read access to all entries except the `userPassword` attribute. The second ACI allows users to update their own contact information. The third ACI allows the `uid=admin` user full access privileges to all user attributes in the `dc=example,dc=com` subtree.

```
aci: (targetattr!="userPassword")(version 3.0; acl "Allow anonymous
   read access for anyone"; allow (read,search,compare) userdn="ldap:///anyone";)
aci: (targetattr="telephonenumber||street||homePhone||l||st")
  (version 3.0; acl "Allow users to update their own contact info";
   allow (write) userdn="ldap:///self";)
aci: (targetattr="*")(version 3.0; acl "Grant full access for the admin user";
  allow (all) userdn="ldap:///uid=admin,dc=example,dc=com";)
```

An important note must be made when assigning access to user and operational attributes, which can be outlined in an example to show the implications of the Directory Proxy Server not distinguishing between these attributes. It can be easy to inadvertently create an access control instruction that grants far more capabilities to a user than originally intended. Consider the following example:

```
aci: (targetattr!="uid||employeeNumber")
  (version 3.0; acl "Allow users to update their own entries";
    allow (write) userdn="ldap:///self";)
```

This instruction is intended to allow a user to update any attribute in his or her own entry with the exception of `uid` and `employeeNumber`. This ACI is a very common type of rule and seems relatively harmless on the surface, but it has very serious consequences for a directory server that does not distinguish between user attributes and operational attributes. It allows users to update operational attributes in their own entries, and could be used for a number of malicious purposes, including:

- A user could alter password policy state attributes to become exempt from password policy restrictions.
- A user could alter resource limit attributes and bypass size limit, time limit, and look-through-limit constraints.
- A user could add access control rules to his or her own entry, which could allow them to make their entry completely invisible to all other users including administrators granted full rights by access control rules, but excluding users with the `bypass-acl` privilege, allow them to edit any other attributes in their own entry including those excluded by rules like `uid` and `employeeNumber` in the example above, or add, modify, or delete any entries below his or her own entry.

Because the UnboundID® Directory Proxy Server does not automatically include operational attributes in the target attribute list, these kinds of ACIs do not present a security risk for it. Also note that users cannot add ACIs to any entries unless they have the `modify-acl` privilege.

Another danger in using the `(targetattr!="x")` pattern is that two ACIs within the same scope could have two different `targetattr` policies that cancel each other out. For example, if one ACI has `(targetattr!="cn||sn")` and a second ACI has `(targetattr!="userPassword")`, then the net effect is `(targetattr="*")`, because the first ACI inherently allows `userPassword`, and the second allows `cn` and `sn`.

targetscope

The `targetscope` keyword is used to restrict the scope of an access control rule. By default, ACIs use a subtree scope, which means that they are applied to the target entry (either as defined by the target clause of the ACI, or the entry in which the ACI is define if it does not include a target), and all entries below it. However, adding the `targetscope` element into an access control rule can restrict the set of entries to which it applies.

The following `targetscope` keyword values are allowed:

- **base**. Indicates that the access control rule should apply only to the target entry and not to any of its subordinates.
- **onelevel**. Indicates that the access control rule should apply only to entries that are the immediate children of the target entry and not to the target entry itself, nor to any subordinates of the immediate children of the target entry.
- **subtree**. Indicates that the access control rule should apply to the target entry and all of its subordinates. This is the default behavior if no `targetscope` is specified.
- **subordinate**. Indicates that the access control rule should apply to all entries below the target entry but not the target entry itself.

The following ACI targets all users to view the operational attributes (`supportedControl`, `supportedExtension`, `supportedFeatures`, `supportedSASLMechanisms`, `vendorName`, and `vendorVersion`) present in the root DSE entry. The `targetscope` is base to limit users to view only those attributes in the root DSE.

```
aci: (target="ldap:///")(targetscope="base")
    (targetattr="supportedControl||supportedExtension||
     supportedFeatures||supportedSASLMechanisms||vendorName||vendorVersion")
    (version 3.0; acl "Allow users to view Root DSE Operational Attributes";
     allow (read,search,compare) userdn="ldap:///anyone")
```

Authentication Method

The `authmethod` keyword can be used to indicate that an access control rule should be applied based on the way in which the user authenticated to the server. The `authmethod` keyword allows the following values:

- **none**. Indicates that the rule should apply to clients that have not attempted to authenticate, that performed an anonymous bind (for example, using simple authentication with a zero-length DN and password), or whose last bind attempt was not successful.
- **simple**. Indicates that the rule should apply to clients that authenticated using simple authentication with a non-empty DN and password.
- **ssl**. Indicates that the rule should apply to clients that authenticated using a client certificate via the SASL EXTERNAL mechanism.
- **sasl <sasl_mechanism>**. Indicates that the rule should apply to clients that authenticated using any SASL mechanism. You can restrict the SASL authentication to a particular mechanism by including its name. For example, “sasl DIGEST-MD5” indicates that the rule should only apply to clients that authenticated using the DIGEST-MD5 SASL mechanism.

Use of Controls and Extended Operations

The UnboundID® Directory Proxy Server adds new keywords that support access control to request controls and extended operations. The `targetcontrol` keyword can be used to indicate whether a given request control may be used. The `extop` keyword can be used to indicate whether a given extended operation can be used. For both keywords, the value is the object identifier for the associated control or extended operation. Multiple OIDs can be provided by separating them with the two pipe characters (optionally surrounded by spaces). For example:

- (targetcontrol="2.16.840.1.113730.3.4.2")
- (extop="1.3.6.1.4.1.4203.1.11.1 || 1.3.6.1.4.1.1466.20037")

Working with Privileges

In addition to the access control implementation, the UnboundID® Directory Proxy Server includes a privilege subsystem that can also be used to control what users are allowed to do. The privilege subsystem works in conjunction with the access control subsystem so that privileged operations are only allowed if they are allowed by the access control configuration and the user has all of the necessary privileges.

Privileges can be used to grant normal users the ability to perform certain tasks that, in most other directories, would only be allowed for the root user. In fact, the capabilities extended to root users in the UnboundID® Directory Proxy Server are all granted through privileges, so you can create a normal user account with the ability to perform some or all of the same actions as root users.

Administrators can also remove privileges from root users so that they are unable to perform certain types of operations. Multiple root users can be defined in the server with different sets of privileges so that the capabilities that they have are restricted to only the tasks that they need to be able to perform.

Available Privileges

The following privileges are defined in the UnboundID® Directory Proxy Server.

Table 7: Summary of Privileges

Privilege	Description
audit-data-security	This privilege is required to initiate a data security audit on the server, which is invoked by the `audit-data-security` tool.
backend-backup	This privilege is required to initiate an online backup through the tasks interface. The server's access control configuration must also allow the user to add the corresponding entry in the tasks backend.
backend-restore	This privilege is required to initiate an online restore through the tasks interface. The server's access control configuration must also allow the user to add the corresponding entry in the tasks backend.

Privilege	Description
bypass-acl	This privilege allows a user to bypass access control evaluation. For a user with this privilege, any access control determination made by the server immediately returns that the operation is allowed. Note, however, that this does not bypass privilege evaluation, so the user must have the appropriate set of additional privileges to be able to perform any privileged operation (for example, a user with the `bypass-acl` privilege but without the `config-read` privilege is not allowed to access the server configuration).
bypass-pw-policy	This privilege allows a user entry to bypass password policy evaluation. This privilege is intended for cases where external synchronization might require passwords that violate the password validation rules. The privilege is not evaluated for bind operations so that password policy evaluation will still occur when binding as a user with this privilege.
bypass-read-acl	This privilege allows the associated user to bypass access control checks performed by the server for bind, search, and compare operations. Access control evaluation may still be enforced for other types of operations.
config-read	This privilege is required for a user to access the server configuration. Access control evaluation is still performed and can be used to restrict the set of configuration objects that the user is allowed to see.
config-write	This privilege is required for a user to alter the server configuration. The user is also required to have the `config-read` privilege. Access control evaluation is still performed and can be used to restrict the set of configuration objects that the user is allowed to alter.
disconnect-client	This privilege is required for a user to request that an existing client connection be terminated. The connection is terminated through the disconnect client task. The server's access control configuration must also allow the user to add the corresponding entry to the tasks backend.
jmx-notify	This privilege is required for a user to subscribe to JMX notifications generated by the directory server. The user is also required to have the `jmx-read` privilege.
jmx-read	This privilege is required for a user to access any information provided by the Directory Proxy Server via the Java Management Extensions (JMX).
jmx-write	This privilege is required for a user to update any information exposed by the Directory Proxy Server via the Java Management Extensions (JMX). The user is also required to have the `jmx-read` privilege. Note that currently all of the information exposed by the server over JMX is read-only.

ldif-export	This privilege is required to initiate an online LDIF export through the tasks interface. The server's access control configuration must also allow the user to add the corresponding entry in the Tasks backend. To allow access to the Tasks backend, you can set up a global ACI that allows access to members of an Administrators group as follows:
	
	`$ 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";)'`

ldif-import	This privilege is required to initiate an online LDIF import through the tasks interface. The server's access control configuration must also allow the user to add the corresponding entry in the Tasks backend. To allow access to the Tasks backend, configure the global ACI as shown in the previous description of the `ldif-export` privilege.
lockdown-mode	This privilege allows the associated user to request that the server enter or leave lockdown mode, or to perform operations while the server is in lockdown mode.
modify-acl	This privilege is required for a user to add, modify, or remove access control rules defined in the server. The server's access control configuration must also allow the user to make the corresponding change to the `aci` operational attribute.

Privilege	Description
password-reset	This privilege is required for one user to be allowed to change another user's password. This privilege is not required for a user to be allowed to change his or her own password. The user must also have the access control instruction privilege to write the `userPassword` attribute to the target entry.
privilege-change	This privilege is required for a user to change the set of privileges assigned to a user, including the set of privileges, which are automatically granted to root users. The server's access control configuration must also allow the user to make the corresponding change to the `ds-privilege-name` operational attribute.
proxied-auth	This privilege is required for a user to request that an operation be performed with an alternate authorization identity. This privilege applies to operations that include the proxied authorization v1 or v2 control operations that include the intermediate client request control with a value set for the client identity field, or for SASL bind requests that can include an authorization identity different from the authentication identity.
server-restart	This privilege is required to initiate a server restart through the tasks interface. The server's access control configuration must also allow the user to add the corresponding entry in the tasks backend.
server-shutdown	This privilege is required to initiate a server shutdown through the tasks interface. The server's access control configuration must also allow the user to add the corresponding entry in the tasks backend.
soft-delete-read	This privilege is required for a user to access a soft-deleted-entry.
stream-values	This privilege is required for a user to perform a stream values extended operation, which obtains all entry DNs and/or all values for one or more attributes for a specified portion of the DIT.
unindexed-search	This privilege is required for a user to be able to perform a search operation in which a reasonable set of candidate entries cannot be determined using the defined index and instead, a significant portion of the database needs to be traversed to identify matching entries. The server's access control configuration must also allow the user to request the search.
update-schema	This privilege is required for a user to modify the server schema. The server's access control configuration must allow the user to update the operational attributes that contain the schema elements.

Privileges Automatically Granted to Root Users

The special abilities that root users have are granted through privileges. Privileges can be assigned to root users in two ways:

- By default, root users may be granted a specified set of privileges. Note that it is possible to create root users which are not automatically granted these privileges by including the `ds-cfg-inherit-default-root-privileges` attribute with a value of FALSE in the entries for those root users.
- Individual root users can have additional privileges granted to them, and/or some automatically-granted privileges may be removed from that user.

The set of privileges that are automatically granted to root users is controlled by the `default-root-privilege-name` property of the Root DN configuration object. By default, this set of privileges includes:

➢ audit-data-security

- backend-backup
- backend-restore
- bypass-acl
- config-read
- config-write
- disconnect-client
- ldif-export
- lockdown-mode
- modify-acl
- password-reset
- privilege-change
- server-restart
- server-shutdown
- soft-delete-read
- stream-values
- unindexed-search
- update-schema

The privileges not granted to root users by default includes:

- bypass-read-acl
- jmx-read
- jmx-write
- jmx-notify
- proxied-auth
- bypass-pw-policy

The set of default root privileges can be altered to add or remove values as necessary. Doing so will require the `config-read`, `config-write`, and `privilege-change` privileges, as well as either the `bypass-acl` privilege or sufficient permission granted by the access control configuration to make the change to the server's configuration.

Assigning Privileges to Normal Users and Individual Root Users

Privileges can be granted to normal users on an individual basis. This can be accomplished by adding the `ds-privilege-name` operational attribute to that user's entry with the names of the desired privileges. For example, the following change will grant the `proxied-auth` privilege to the `uid=proxy,dc=example,dc=com` account:

```
dn: uid=proxy,dc=example,dc=com
changetype: modify
add: ds-privilege-name
ds-privilege-name: proxied-auth
```

The user making this change will be required to have the `privilege-change` privilege, and the server's access control configuration must also allow the requester to write to the `ds-privilege-name` attribute in the target user's entry.

This same method can be used to grant privileges to root users that they would not otherwise have through the set of default root privileges. You can also remove default root privileges from

root users by prefixing the name of the privilege to remove with a minus sign. For example, the following change grants a root user the `jmx-read` privilege in addition to the set of default root privileges, and removes the `server-restart` and `server-shutdown` privileges:

```
dn: cn=Sync Root User,cn=Root DNs,cn=config
changetype: modify
add: ds-privilege-name
ds-privilege-name: jmx-read
ds-privilege-name: -server-restart
ds-privilege-name: -server-shutdown
```

Note that because root user entries exist in the configuration, this update requires the `config-read` and `config-write` privileges in addition to the `privilege-change` privilege.

Disabling Privileges

Although the privilege subsystem in the UnboundID® Directory Proxy Server is a very powerful feature, it might break some applications if they expect to perform some operation that requires a privilege that they do not have. In the vast majority of these cases, you can work around the problem by simply assigning the necessary privilege manually to the account used by that application. However, if this workaround is not sufficient, or if you need to remove a particular privilege (for example, to allow anyone to access information via JMX without requiring the `jmx-read` privilege), then privileges can be disabled on an individual basis.

The set of disabled privileges is controlled by the `disabled-privilege` property in the global configuration object. By default, no privileges are disabled. If a privilege is disabled, then the server behaves as if all users have that privilege.

Chapter 6 Managing Entry-Balancing Replication

Replication in the UnboundID® Directory Proxy Server synchronizes directory data between all servers in the topology. In a deployment using the entry-balancing feature, however, directory data under the entry-balancing point is split into multiple data sets. Each data set is replicated to ensure high availability between a subset of the servers in the topology. Other directory data, such as the schema or data above the entry-balancing point, is replicated between all servers in the topology.

This chapter presents the following information about replication in an entry-balancing environment:

Topics:

Overview of Replication in an Entry-Balancing Environment

In an entry-balanced deployment, some data is replicated everywhere, such as the schema, the server registry, and other shared data, and some data is replicated only on certain servers. A replication domain contains all of the servers in a replicated topology and shares a schema. The replication domain is associated with the base DN and must be a base DN of a backend.

By default, replication propagates updates to all replication servers in the topology. Updates to data under the entry-balancing point, however, must be replicated only among server instances in the same data set. Replication requires that, in such deployments, the directory server is configured with a replication set name global configuration property, and two backends. One backend has a base DN that is replicated globally (such as `dc=example,dc=com`) and the second backend has a base DN associated with the entry-balancing point (such as `ou=people,dc=example,dc=com`).

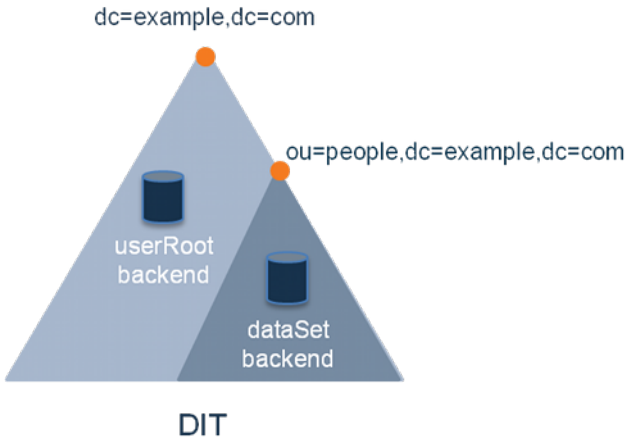


Figure 6: Global and Restricted Backends

If a data set name is not defined when you set up the proxy server, one will be provided by default. The proper configuration of an entry-balancing environment requires coordination between the directory server and proxy server. Once replication is enabled, the replication domain may be designated as the domain participating in entry balancing.

Replication Prerequisites in an Entry-Balancing Deployment

Replication in an entry-balanced deployment requires the following:

- **Multiple local DB backends**. When you set up the directory instances, you need two backends, a global backend for globally replicated data, such as userRoot, and a restricted backend for the balancing point base DN, dataSet. Both backends, the global and the restricted, need to be initialized separately. Depending on the size of the dataset, you can initialize the backends using the `dsreplication` tool, the `import-ldif` tool, or binary copy.

See the UnboundID® Directory Proxy Server Administration Guide for more information about initializing replication data.

Note: When using the binary copy method to initialize a new server, the `replicationChanges` backend should not be copied across servers taking part in different restricted sets. Instead, allow the `replicationChanges` database to be automatically updated by the replication server, when the new server is enabled for replication.

- **Replication set name**. Every directory server in your replicated topology must have a replication set name. This replication set name coordinates the proxy server and the directory server. The restricted domain is only replicated within instances using the same replication set name.
- **Multiple proxy server subtree views**. The entry-balanced proxy configuration relies on multiple subtree views, one for the globally replicated base DN and one for the entry-balancing point base DN. The globally replicated base DN will have a proxying request processor associated with it. The restricted base DN will have an entry-balancing request processor associated with it. This configuration is best achieved using the `create-initial-proxy-config` tool after running setup.

About the --restricted Argument of the dsreplication Command-Line Tool

When enabling replication for a server that takes part in an entry balanced environment, it is recommended that the multiple domains involved are enabled at the same time. There is a global domain, and a restricted domain, where the restricted domain represents the entry-balancing point. Each base DN is defined in a separate Local DB Backend. The `dsreplication` CLI tool has a `--restricted` argument that is used to specify which base DN is considered an entry balancing point.

To Use the --restricted Argument of the dsreplication Command-Line Tool

- Run `dsreplication` to enable replication between two servers with entry balancing.
  - You can run the command in non-interactive mode as follows:

```
$ bin/dsreplication enable --host1 host1.example.com \
  --port1 1389 --bindDN1 "cn=Directory Manager" \
  --bindPassword1 secret --replicationPort1 8989 \
  --host2 host2.example.com --port2 2389 \
  --bindDN2 "cn=Directory Manager" --bindPassword2 secret \
  --replicationPort2 8989 --baseDN dc=example,dc=com \
  --baseDN ou=people,dc=example,dc=com \
  --restricted ou=people,dc=example,dc=com
```

  - Alternatively, you can enable replication using the interactive command line, making sure to specify that an entry balancing is being used and specifying the base DN of the entry-balancing point. After entering `dsreplication` and entering the LDAP connection parameters, follow the prompts presented.

```
You must choose at least one base DN to be replicated.

Replicate base DN dc=example,dc=com? (yes / no) [yes]: yes

Replicate base DN ou=people,dc=example,dc=com? (yes / no) [yes]: yes

Do you plan to configure entry balancing using the Proxy Server? (yes / no) [no]:
 yes

Is dc=example,dc=com an entry-balancing point? (yes / no) [no]: no

Is ou=people,dc=example,dc=com an entry-balancing point? (yes / no) [no]: yes
```

Checking the Status of Replication in an Entry-Balancing Deployment

You can use the `dsreplication` status tool to check the status of an entry-balancing deployment. In this example, the `ou=people,dc=example,dc=com` subtree is entry-balanced. The data is split into two sets, `set1` and `set2`. The servers `host1` and `host2` are in replication set `set1` and servers `host3` and `host4` are in replication set `set2`.

To Check the Status of Replication in an Entry-Balancing Deployment

- Run the `dsreplication` command to get a status of replication in the entry-balancing deployment. To view a specific set, use the `--setName` option to see only the specific replication set; otherwise, all of the sets will be displayed by default.

```
$ bin/dsreplication status --hostname host1.example.com \
  --port 1389 --adminUID admin --adminPassword secret
```

```
          --- Replication Status for dc=example,dc=com: Enabled ---

Server    : Entries: Replication Backlog: Oldest Backlog Change Age : Port :Security

----------:--------:--------------------:--------------------------:------:-------

austin1.example.com:1389 : 1000    : 0            : N/A             : 8989 : Enabled
austin2.example.com:2389 : 1000    : 0            : N/A             : 8989 : Enabled
newyork1.example.com:3389: 1000    : 0            : N/A             : 8989 : Enabled
newyork2.example.com:4389: 1000    : 0            : N/A             : 8989 : Enabled


 -- Replication Status for ou=people,dc= example,dc=com (Set: set1): Enabled --

Server    : Entries: Replication Backlog: Oldest Backlog Change Age : Port :Security

----------:--------:--------------------:--------------------------:------:-------

austin1.example.com:1389 : 1000000 : 0            : N/A           : 8989 : Enabled
austin2.example.com:2389 : 1000000 : 0            : N/A           : 8989 : Enabled


---Replication Status for ou=people,dc= example,dc=com (Set: set2): Enabled ---

Server    : Entries: Replication Backlog: Oldest Backlog Change Age : Port :Security

----------:--------:--------------------:--------------------------:------:-------

newyork1.example.com:3389 : 1000000 : 0            : N/A           : 8989 : Enabled
newyork2.example.com:4389 : 1000000 : 0            : N/A           : 8989 : Enabled
```

Example of Configuring Entry-Balancing Replication

This section describes how to set up a four-server multi-master replication topology that uses entry balancing to distribute entries across the servers. The procedure assumes that none of the servers have participated in any previous replication topology. Also, this procedure uses the binary copy method for initializing data into the replicas, which is recommended for production environments.

Assumptions

The following procedures assume that all directory server instances are built from scratch and are solely dedicated as directory servers. The example uses the LDAP (389) and replication (8989) ports respectively. It configures the following hosts:

- austin1.example.com
- newyork1.example.com
- austin2.example.com
- newyork2.example.com

In this example, we have an entry-balancing domain of `dc=example,dc=com`. The data below the entry balancing point of `ou=people,dc=example,dc=com` is distributed across two data sets, dataSet1 and dataSet2. Each data set is replicated between two directory servers. Each of these servers is associated with one of two locations, Austin and New York.

The following graphic illustrates this basic entry-balancing deployment:

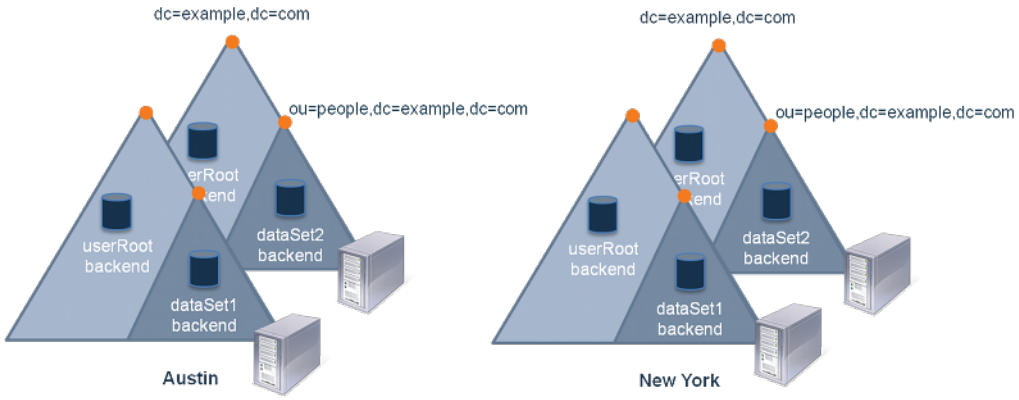


Figure 7: Basic Entry-Balancing Deployment

Replication takes place as described in the following graphic. Data in the userRoot backend is replicated between all of the servers in both locations. Data in each data set is replicated between a server in each location:

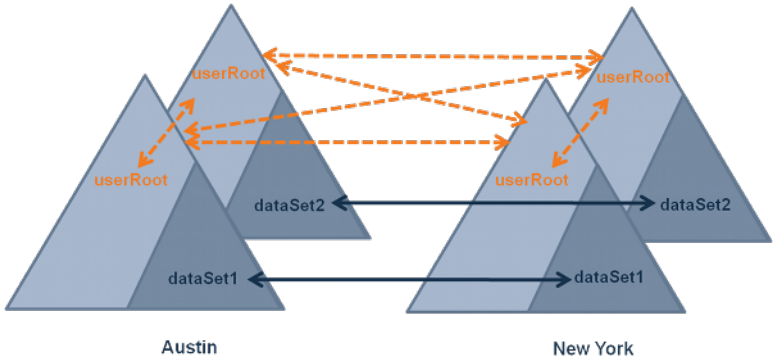


Figure 8: Replication Flow in the Basic Entry-Balancing Deployment

Configuration Summary

To configure multi-mastered replication in an entry-balanced deployment, you must do the following:

- Install the directory server instances. In this deployment example, we will install two directory servers in the Austin location and two directory servers in the New York location.
- Create a new dataset backend and set the replication set name for each directory server instance.
- Create and set locations for the Austin and New York servers.
- Import or create data for the non-entry balanced global domain and for the entry-balanced backends.
- Configure replication.
- Configure the proxy servers.
- Check the status of replication.

To Install the Directory Servers

First, install the directory server instances. In this example, we install the following four servers, two in the Austin location and two in the New York location:

- austin1.example.com
- austin2.example.com
- newyork1.example.com
- newyork2.example.com

1. We install the first server, austin1, as follows:

```
root@austin1# ./setup --cli --baseDN dc=example,dc=com \
--ldapPort 389 --rootUserDN "cn=Directory Manager" \
--rootUserPassword pass --no-prompt --acceptLicense
```

2. Install the second Austin server, austin2, in the same way:

```
root@austin2 # ./setup --cli --baseDN dc=example,dc=com \
--ldapPort 389 --rootUserDN "cn=Directory Manager" \
--rootUserPassword pass --no-prompt --acceptLicense
```

3. Next, install the two New York servers, newyork1 and newyork2, as follows:

```
root@newyork1# ./setup --cli --baseDN dc=example,dc=com \
--ldapPort 389 --rootUserDN "cn=Directory Manager" \
--rootUserPassword pass --no-prompt --acceptLicense

root@newyork# ./setup --cli --baseDN dc=example,dc=com \
--ldapPort 389 --rootUserDN "cn=Directory Manager" \
--rootUserPassword pass --no-prompt --acceptLicense
```

To Create the Database Backends

We create a dataset database backends on each Austin directory server instance.

1. First, create dataset on austin1.example.com:

```
root@austin1# ./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. Create dataset on austin2.example.com:

```
root@austin2# ./bin/dsconfig --no-prompt create-backend \
--backend-name dataset --type local-db --set enabled:true \
--set base-dn:ou=people,dc=example,dc=com
```

3. Set the replication set name for austin1.example.com to dataset1:

```
 root@austin1# ./bin/dsconfig --no-prompt \
set-global-configuration-prop \
--set replication-set-name:dataset1
```

4. Set the replication set name for austin2.example.com to dataset2:

```
 root@austin2# ./bin/dsconfig --no-prompt \
set-global-configuration-prop \
--set replication-set-name:dataset2
```

5. Create a dataset database backends on each of the New York directory server instances in the same way. We create dataset on the newyork1.example.com and newyork1.example.com instances:

```
root@newyork1# ./bin/dsconfig --no-prompt create-backend \
--backend-name dataset --type local-db --set enabled:true \
--set base-dn:ou=people,dc=example,dc=com

root@newyork2# ./bin/dsconfig --no-prompt create-backend \
--backend-name dataset --type local-db --set enabled:true \
--set base-dn:ou=people,dc=example,dc=com
```

6. Set the replication set name for newyork1.example.com to dataset1:

```
root@newyork1# ./bin/dsconfig --no-prompt \
set-global-configuration-prop \
--set replication-set-name:dataset1
```

7. Set the replication set name for newyork2.example.com to dataset2:

```
root@newyork2# ./bin/dsconfig --no-prompt \
set-global-configuration-prop \
--set replication-set-name:dataset2
```

To Create and Set the Locations

1. On the austin1 instance, we create the two locations, newyork and austin, and set the location of this instance to austin:

```
root@austin1# ./bin/dsconfig --no-prompt create-location --location-name austin

root@austin1# ./bin/dsconfig --no-prompt create-location --location-name newyork \
--set preferred-failover-location:austin

root@austin1# ./bin/dsconfig --no-prompt set-location-prop --location-name austin \
--add preferred-failover-location:newyork

root@austin1# ./bin/dsconfig --no-prompt set-global-configuration-prop \
--set location:austin
```

2. We do the same on the austin2 instance, creating the two locations, newyork and austin, and setting the location of the instance to austin:

```
root@austin2# ./bin/dsconfig --no-prompt create-location \
--location-name austin

root@austin2# ./bin/dsconfig --no-prompt create-location \
--location-name newyork \
--set preferred-failover-location:austin

root@austin2# ./bin/dsconfig --no-prompt set-location-prop \
--location-name austin \
--add preferred-failover-location:newyork

root@austin2# ./bin/dsconfig --no-prompt set-global-configuration-prop \
--set location:austin
```

3. Configure the two locations on the newyork1 instance, and setting its location to newyork:

```
root@newyork1# ./bin/dsconfig --no-prompt create-location \
--location-name austin

root@newyork1# ./bin/dsconfig --no-prompt create-location \
--location-name newyork \
--set preferred-failover-location:austin

root@newyork1# ./bin/dsconfig --no-prompt set-location-prop \
--location-name austin \
--add preferred-failover-location:newyork

root@newyork1# ./bin/dsconfig --no-prompt set-global-configuration-prop \
--set location:newyork
```

4. Configure the two locations on the newyork2 instance, also setting its location to newyork:

```
root@newyork2# ./bin/dsconfig --no-prompt create-location \
--location-name austin

root@newyork2# ./bin/dsconfig --no-prompt create-location \
--location-name newyork \
-set preferred-failover-location:austin

root@newyork2# ./bin/dsconfig --no-prompt set-location-prop \
--location-name austin \
--add preferred-failover-location:newyork
```

```
root@newyork2# ./bin/dsconfig --no-prompt \
set-global-configuration-prop --set location:newyork
```

To Import the Entries

We import the userRoot data, based on data defined in the `userRoot.ldif` file, into each of the servers in our topology. This file does not contain entries at or within the entry-balancing point, `ou=people,dc=example,dc=com`.

1. Use the `import-ldif` command to import the userRoot data.

```
root@austin1# ./bin/import-ldif --backendID userRoot \
--ldifFile /data/userRoot.ldif \
--includeBranch dc=example,dc=com \
--rejectFile /data/austin1-import-rejects \
--port 389
--hostname austin1.example.com
```

2. Import the dataset1 data into the servers in that replication set, austin1 and newyork1, as follows:

```
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

root@newyork1# ./bin/import-ldif --backendID dataset \
--ldifFile /data/dataset1.ldif \
--includeBranch ou=people,dc=example,dc=com \
--rejectFile /data/newyork1-dataset-import-rejects \
--hostname newyork1.example.com --port 389
```

3. Finally, we import the dataset2 data into the servers in the set2 replication set, austin2 and newyork2, as follows:

```
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

root@newyork2# ./bin/import-ldif --backendID dataset \
--ldifFile /data/dataset2.ldif \
--includeBranch ou=people,dc=example,dc=com \
--rejectFile /data/newyork2-dataset-import-rejects \
--hostname newyork2.example.com --port 389
```

To Enable Replication in an Entry-Balancing Deployment

Now we can enable replication between the servers in our topology. First, we configure replication between austin1 and austin2. Notice that we specify the `--restricted` domain in the `dsreplication` command even though these two servers are not sharing the same dataset.

1. Run `dsreplication enable` to enable the servers in the topology.

```
root@austin1# ./bin/dsreplication enable \
--host1 austin1.example.com \
--port1 389 --bindDN1 "cn=directory manager" \
--bindPassword1 pass --host2 austin2.example.com \
--port2 389 --bindDN2 "cn=directory manager" \
--bindPassword2 pass \
--replicationPort1 8989 \
```

```
--replicationPort2 8989 \
--baseDN dc=example,dc=com \
--baseDN ou=people,dc=example,dc=com \
--restricted ou=people,dc=example,dc=com \
--adminUID admin --adminPassword pass --trustAll \
--no-prompt
```

2. Enable replication between austin1 and newyork1. This procedure automatically enables replication between austin2 and newyork1 as well.

```
root@austin1# ./bin/dsreplication enable \
--host1 austin1.example.com \
--port1 389 --bindDN1 "cn=directory manager" \
--bindPassword1 pass --host2 newyork1.example.com \
--port2 389 --bindDN2 "cn=directory manager" \
--bindPassword2 pass \
--replicationPort1 8989 \
--replicationPort2 8989 \
--baseDN dc=example,dc=com \
--baseDN ou=people,dc=example,dc=com \
--restricted ou=people,dc=example,dc=com \
--adminUID admin --adminPassword pass --trustAll \
--no-prompt
```

3. Finally, configure replication between austin1 and newyork2. Because of the replication agreements created in the previous step, this procedure automatically enables replication between austin2 and newyork2 as well.

```
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
```

To Check the Status of Replication

Once replication has been configured, check the status of the replication topology using the `dsreplication status` command.

- Run the `dsreplication status` command to check its status.

```
root@austin1# ./bin/dsreplication status \
--adminPassword pass --no-prompt --port 389
```

Detaching a Replication Set from a Topology

Sometimes, you may need to detach a single server or a set of servers from the replication topology without disabling replication on the servers. To do so, you can use the `dsreplication detach` command. After you run the command, replication no longer occurs between replication sets 1 and 2 (shown below), though replication continues to take place within each of the two sets:

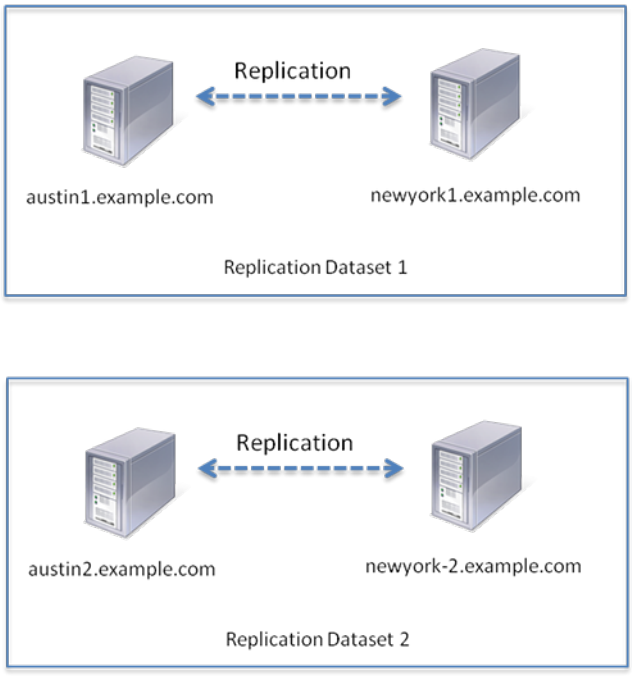


Figure 9: Replication Datasets

To Detach a Replication Set from a Topology

Run the `dsreplication detach` command to detach a replication set from the topology.

```
root@austin1# ./bin/dsreplication detach \
--hostname austin1.example.com --port 1389 --adminUID admin \
--adminPassword password --setName dataset1
```

Detaching a Server from a Replication Set

You can also detach a single server from the replication topology using the `dsreplication detach` command and specifying the hostname of the server that you want to detach. Use the `detach` subcommand only if austin1 will be added back to the replication topology. Otherwise, consider using the `disable` subcommand instead. For example, you might detach a single server for a short time to diagnose a problem without taking the server out of the topology. Once re-enabled, it receives updates from the other replication servers if the purge delay has not been exceeded. Otherwise, you need to reinitialize the re-enabled replica.

For example, if you detach one server, austin1, from a replication set, austin1 still has replication enabled even though it no longer participates in the replication topology of `newyork1`, `austin2`, and `newyork2`. Updates made on `austin1` will be recorded in the changelog database of the embedded replication server instance. When `austin1` is added back to the topology, these updates will be propagated to the other servers. Changes made on the other servers while `austin1` was detached will also be applied.

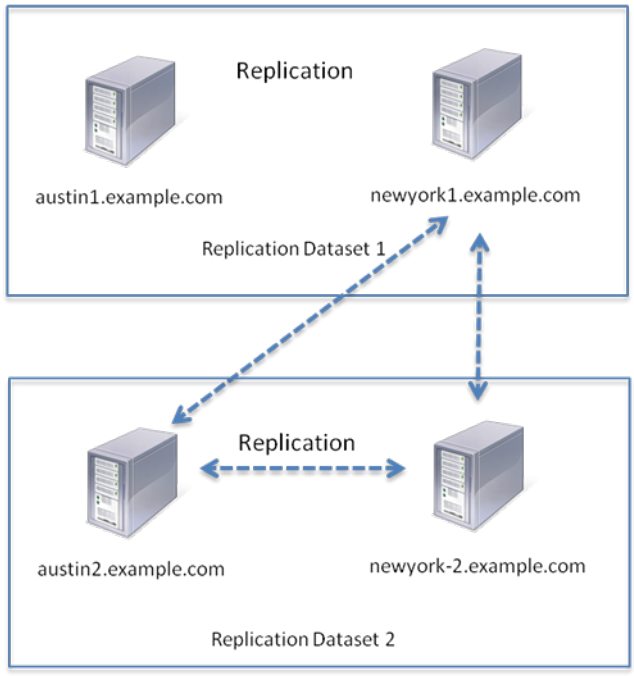


Figure 10: Detached Server from a Replication Set

To Detach a Single Server from a Replication Set

- Run `dsreplication detach` to detach the server from a replication set.

```
$ bin/dsreplication detach --hostname austin1.example.com \
  --port 1389 --adminUID admin --adminPassword password
```

Chapter 7 Deploying the Directory Proxy Server

You can deploy UnboundID® Directory Proxy Server in a variety of ways, depending upon the needs of your enterprise. This chapter describes and illustrates several common deployment scenarios.

It includes the following sections:

Topics:

Creating a Standard Multi-Location Deployment

In this example deployment, UnboundID® Directory Proxy Server will be deployed in the data centers of two geographic locations: east and west. All LDAP external servers in this deployment are UnboundID® Directory Servers. The directory servers in the eastern city are assigned to the location named east, and the directory servers in the western city are assigned to the location named west.

This example refers to four directory servers in two locations with replication of the `dc=example,dc=com` base DN enabled:

- ds-east-01.example.com
- ds-east-02.example.com
- ds-west-01.example.com
- ds-west-01.example.com

We will configure four proxy servers:

- proxy-east-01.example.com
- proxy-east-02.example.com
- proxy-west-01.example.com
- proxy-west-02.example.com

Overview of the Deployment Steps

In this deployment scenario, we will take the following steps:

- Install the first Directory Proxy Server in east location using the `setup` or `setup.bat` file included in the zip installation file.
- Use the `create-initial-proxy-config` tool to provide a proxy user bind DN and password, define locations for each of our data centers, and configure the LDAP external servers in these data centers.
- Test external server communications after initial setup is complete and test a simulated external server failure.
- Install the second directory 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 proxy using the configuration cloning feature.
- Install two proxy servers in the west location, which includes using the setup file and manually setting the location to west using the `dsconfig` command, as well as copying the configuration of the proxy server using the configuration cloning feature.

After the directory proxy server has been configured and tested, we then provide a tour of the configuration of each of the directory proxy server components. These properties can be modified later as needed using the `dsconfig` tool.

Installing the First Directory Proxy Server

To begin with, we have the UnboundID® Directory Proxy Server installation zip file. In this example, we plan to use SSL security, so we also have a keystore certificate database and a pin file that contains the private key password for the keystore. The keystore files are only necessary when using SSL or StartTLS.

In this deployment scenario, the keystore database is assumed to be a Java Key Store (JKS), which can be created by the keytool program. For more information about using the keytool, see the "Security Chapter" in the UnboundID® Directory Server Administration Guide.

The UnboundID-Proxy directory contains the following:

```
root@proxy-east-01: ls
ExampleKeystore.jks   ExampleTruststore.jks ExampleKeystore.pin
UnboundID-Proxy-3.6.0.0-with-je.zip
```

The `ExampleKeystore.jks` keystore file contains the private key entry for the `proxy-east-01.example.com` server certificate with the alias `server-cert`. The server certificate, CA, and intermediate signing certificates are all contained in the `ExampleTruststore.jks` file. The password for `ExampleKeystore.jks` is defined in clear text in the corresponding pin file, though the name of the file need not match as it does in our example. The private key password in our example is the same as the password defined for the `ExampleKeystore.jks` keystore.

To Install the First Directory Proxy Server

1. Unzip the compressed archive file into the UnboundID-Proxy directory and move to this directory.

   ```
   root@proxy-east-01: unzip -q UnboundID-Proxy-<version>-with-je.zip
   root@proxy-east-01: cd UnboundID-Proxy
   ```

2. Because we are configuring SSL security, copy the keystore and pin files into the `config` directory.

   ```
   root@proxy-east01: cp ../*Keystore* config/
   root@proxy-east01: cp ../*Truststore* config/
   ```

3. Next, we install the first directory proxy server by running the `setup` tool on `proxy-east-01.example.com` as follows:

   ```
   root@proxy-east01: ./setup --no-prompt --acceptLicense \
   --ldapPort 389 --rootUserPassword pass \
   --aggressiveJVMTuning --maxHeapSize 1g \
   --enableStartTLS --ldapsPort 636 \
   --useJavaKeystore config/ExampleKeystore.jks \
   --keyStorePasswordFile config/ExampleKeystore.pin \
   --certNickname server-cert \
   --useJavaTrustStore config/ExampleTruststore.jks
   ```

   New keystore password files are created in `config/keystore.pin`. The original file, `config/ExampleKeystore.pin`, is no longer needed.

4. If you are not using SSL or StartTLS, then the SSL arguments are not necessary as follows:

```
root@proxy-east01: ./setup --no-prompt --acceptLicense \
--ldapPort 389 --rootUserPassword pass \
--aggressiveJVMTuning --maxHeapSize 1g
```

Once you have installed the Directory Proxy Server, you can configure it using the `create-initial-proxy-config` tool as presented in the next section.

Configuring the First Directory Proxy Server

Once the Directory Proxy Server has been installed, it can be automatically configured using the `create-initial-proxy-config` tool. This tool can only be used once for this initial configuration, after which we will have to use `dsconfig` to make any changes to our directory proxy server configuration.

Configuring the Directory Proxy Server with the `create-initial-proxy-config` tool involves the following steps:

- Providing a directory proxy server base DN and password.
- Defining locations for each of our data centers, east and west.
- Configuring the LDAP external server in the east location.
- Configuring the LDAP external servers in the west location.
- Applying the changes to the directory proxy server.

To Configure the First Directory Proxy Server

1. Once we have completed setup, we run the `create-initial-proxy-config` tool as follows:

```
root@proxy-east01: bin/create-initial-proxy-config

>>>> >>>> Initial Proxy Configuration Tool

This tool can be used to generate a basic Directory Proxy Server
configuration by prompting for basic information about your topology
including directory server instances, their locations, and credentials
for communicating with them. This tool will record the configuration in a
'dsconfig' batch file and optionally allow you to apply the configuration
to the local Directory Proxy Server.

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 UnboundID, Alcatel-Lucent 8661, or Sun Java
System 5.x, 6.x, or 7.x, or Red Hat (including Fedora and 389)
directory servers

If your topology does not have these characteristics you can use this
tool to define a basic configuration and then use the 'dsconfig' tool
or the web console to fine tune the configuration.

Would you like to continue? (yes / no) [yes]:
```

2. Next, provide the bind DN and password that the directory proxy server will use to authenticate to the backend directory servers. 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 of our directory proxy servers 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.

```
>>>> >>>> External Server Access Credentials

Provide the DN of the root account that will be used when
communicating with Directory Server instances. In a later step
you will be given the opportunity to have this tool create this
account or set its password for each Directory Server instance
you specify.

Enter the DN of the proxy user account [cn=Proxy User,cn=Root,cn=config]:

Enter the password for 'cn=Proxy User,cn=Root,cn=config':

Confirm the password for 'cn=Proxy User,cn=Root,cn=config':
```

3. Next, specify the type of external server communication security that will be used to communicate with the directory servers. For this example, enter the option for 'None'.

```
>>>> >>>> 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]:
```

4. Specify the base DNs of the directory server instances that the Directory Proxy Server will access. For this example, use `dc=example,dc=com`.

```
>>>> >>>> Proxy Base DNs

Enter the base DNs of the Directory Server instances that will
be accessed through the Directory Proxy Server

    b)  back
    q)  quit

Enter a DN or choose a menu item [dc=example,dc=com]:

Are entries within '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]:
```

5. Next, enter any other base DNs of the directory server instances that will be accessed through the directory proxy server. Because we are only using one proxy base DN, press **Enter** in this example.

```
>>>> >>>> Proxy Base DNs

Enter the base DNs 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]:
```

Defining Locations

Next, we define our first location, east, to accommodate the servers in our deployment located on the East Coast of the United States.

To Define Proxy Locations

1. Continuing from the same `create-initial-proxy-config` session, enter a location name for the directory proxy servers. In this example, enter `east`, and then press **Enter**.

```
>>>> >>>> Define Locations

Locations are used to define collections of servers which may
have similar performance characteristics when accessed from
this Directory Proxy Server. For example, a separate Location may be
defined for each data center.

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: east

    1)  Remove east

    b)  back
    q)  quit
```

2. Define a location named `west` for the servers in our deployment located on the West Coast. Then we press **Enter** to indicate that we have finished specifying locations.

```
Enter another location name or choose a menu item
[Press ENTER when finished entering locations]: west

    1)  Remove east
    2)  Remove west

    b)  back
    q)  quit

Enter another location name or choose a menu item [Press ENTER
when finished entering locations]:
```

3. Select the location that contains the directory proxy server itself. Our directory proxy server is located in the east, so press **Enter**.

```
>>>> >>>> Proxy Location

Choose the location for this Directory Proxy Server

    1)  east
    2)  west

    b)  back
    q)  quit

Enter choice [1]:
```

Configuring the External Servers in the East Location

Once the locations have been defined, we need to identify the directory servers. First, we define one of the servers in the east location.

To Configure the External Servers in the East Location

1. Define one of the servers in the east location by entering the host name and port of the server. For this example, enter `ds-east-01.example.com:389`.

```
>>>> External Servers

External Servers identify directory server instances including
host, port, and authentication information.

Enter the host and port (host:port) of the first directory server
in 'east'

    b)  back
    q)  quit

Enter a host:port or choose a menu item [localhost:389]: ds-east-01.example.com:389
```

2. Next, enter the option to prepare the server and all subsequent servers. Preparing the servers involves testing the connections to these servers and sets up the `cn=Proxy User` account on the directory proxy server.

```
Would you like to prepare ds-east-01.example.com:389 for access
by the Proxy Server?

    1)  Yes
    2)  No
    3)  Yes, and all subsequent servers
    4)  No, and all subsequent servers

Enter choice [1]: 3

Testing connection to ds-east-01.example.com:389 ..... Done
Testing 'cn=Proxy User,cn=Root DNs,cn=config' access to
  ds-east-01.example.com:389 .....Denied

Would you like to create or modify root user 'cn=Proxy User'
so that it is available for this 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
Verifying backend 'dc=example,dc=com' ..... Done
```

3. Next, 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`.

```
Enter the DN of an account on ds-east-01.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':

Testing 'cn=Proxy User,cn=Root DNs,cn=config' privileges...Done
Verifying backend 'dc=example,dc=com' .....Done
```

4. We repeat the process to prepare the other server in the east location, ds-east-02.example.com.

```
Enter another server in east or choose a menu item
    1)  Remove ds-east-01.example.com:389

    b)  back
    q)  quit

Enter a host:port or choose a menu item [Press ENTER when finished
entering servers for 'east']: ds-east-02.example.com:389

Testing connection to ds-east-02.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' so
that it is available for this 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
Verifying backend 'dc=example,dc=com' ..... Done
```

5. Press **Enter** to complete preparing the servers.

```
Enter another server in east or choose a menu item

    1)  Remove ds-east-01.example.com:389
    2)  Remove ds-east-02.example.com:389

    b)  back
    q)  quit

Enter a host:port or choose a menu item [Press ENTER when
finished entering servers for 'east']:
```

To Configure the External Servers in the West Location

We repeat the process used for the east location to define the LDAP external servers used for the west location. We define the first external server, ds-west-01.example.com.

1. Define the first external server, ds-west-01.example.com.

```
>>>> >>>> Location 'west' 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 'west'

    b)  back
    q)  quit

Enter a host:port or choose a menu item [localhost:389]: ds-west-01.example.com:389


Testing connection to ds-west-01.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' so that it is available
for this 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
Verifying backend 'dc=example,dc=com' ..... Done
```

2. Define the second server in the west location, ds-west-02.example.com.

```
Enter another server in 'west'
    1)  Remove ds-west-01.example.com:389

    b)  back
    q)  quit

Enter a host:port or choose a menu item [Press ENTER when
finished entering servers]:ds-west-02.example.com:389

Testing connection to ds-west-02.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' so
that it is available for this 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
Verifying backend 'dc=example,dc=com' ..... Done
```

3. Press **Enter** to indicate that we have finished defining external servers for the west location.

```
Enter another server in 'west'

    1)  Remove ds-west-01.example.com:389
    2)  Remove ds-west-02.example.com:389

    b)  back
    q)  quit

Enter a host:port or choose a menu item [Press ENTER when
finished entering servers]:
```

Apply the Configuration to the Directory Proxy Server

Next, we review the configuration summary. Once we have confirmed that the changes are correct, we press **Enter** to write the configuration.

To Apply the Changes to the Directory Proxy Server

1. During the configuration process, the `create-initial-proxy-config` tool writes the configuration settings to a `dsconfig` batch file, which will then be applied to the directory proxy server. The batch file can be reused to configure other servers. On the final step, the `create-initial-proxy-config` tool presents a configuration summary. Review the configuration and then apply the changes to the directory proxy server. Press **Enter** to write the configuration to the server.

```
>>>> >>>> Configuration Summary

External Server Security: None
Proxy User DN:            cn=Proxy User,cn=Root DNs,cn=config

Location east
 Failover Order: west
 Servers: ds-east-01.example.com:389,ds-east-02.example.com:389

Location west
 Failover Order: east
 Servers: ds-west-01.example.com:389,ds-west-02.example.com:389

 Base DN: dc=example,dc=com
   Servers: ds-east-01.example.com:389,ds-east-02.example.com:389,
   ds-west-01.example.com:389, ds-west-02.example.com:389

   b)  back
   q)  quit
```

```
   w)  write configuration

Enter choice [w]:
```

2. On the final confirmation prompt, press **Enter** to apply the changes to the directory proxy server, and then enter the LDAP connection parameters to the server. Once the changes have been applied, we cannot use the `create-initial-proxy-config` tool to configure this directory proxy server again.

```
>>>>>>>> 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
UnboundID-Proxy/dps-cfg.txt .....Done

This tool can apply the configuration changes to the local
Directory Proxy Server. This requires any configured Server SDK
extensions to be in place. Do you want to do this? (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 Processors for dc=example,dc=com ..... Done
Creating Subtree Views for dc=example,dc=com ..... Done
Updating Client Connection Policy for dc=example,dc=com....Done

See /UnboundID-Proxy/logs/create-initial-proxy-config.log
for a detailed log of this operation

To see basic server configuration status and configuration you can launch
UnboundID-Proxy/bin/status
```

Configuring Additional Directory Proxy Servers

We install and configure the second directory proxy server by running the `setup` tool on proxy-east-02.example.com.

To Configure Additional Directory Proxy Servers

1. Copy the keystore and pin files into the `config` directory for the proxy-east-02.example.com server.

```
root@proxy-east-02: cp ../*Keystore* config/
root@proxy-east-02: cp ../*Truststore* config/
```

2. Install the second directory proxy server by running the `setup` tool on proxy-east-02.example.com as follows:

```
root@proxy-east-02: ./setup --no-prompt \
--listenAddress proxy-east-02.example.com \
--ldapPort 389 --enableStartTLS --ldapsPort 636 \
--useJavaKeystore config/ExampleKeystore.jks \
--keyStorePasswordFile config/ExampleKeystore.pin \
--certNickName server-cert \
--useJavaTrustStore config/ExampleTruststore.jks \
--rootUserPassword pass --acceptLicense \
--aggressiveJVMTuning --maxHeapSize 1g \
--localHostName proxy-east-02.example.com \
--peerHostName proxy-east-01.example.com \
--peerPort 389 --location east

UnboundID® Directory Proxy Server <version>
Please wait while the setup program initializes...
Reading Peer Configuration ..... Done
Applying Peer Configuration ..... Done
Updating Administration Data ..... Done
Configuring Directory Proxy Server ..... Done
Configuring Certificates ..... Done
Updating Topology Admin Data ..... Done
Starting Directory Proxy Server ..... Done

To see basic server configuration status and configuration you can
launch /bin/status

See /logs/tools/setup.log for a detailed log of this operation.
```

3. Configure the third directory proxy server, proxy-west-01.example.com in the same way as shown in the previous step. First, copy the keystore and pin files into the `config` directory.

```
root@proxy-west-01: cp ../*Keystore* config/
root@proxy-west-01: cp ../*Truststore* config/
```

4. Run the `setup` tool on proxy-west-01.example.com as follows:

```
root@proxy-west-01: ./setup --no-prompt \
--listenAddress proxy-west-01.example.com \
--ldapPort 389 --enableStartTLS --ldapsPort 636 \
--useJavaKeystore config/ExampleKeystore.jks \
--keyStorePasswordFile config/ExampleKeystore.pin \
--certNickName server-cert \
--useJavaTrustStore config/ExampleTruststore.jks \
--rootUserPassword pass --acceptLicense \
--aggressiveJVMTuning --maxHeapSize 1g \
--localHostName proxy-west-01.example.com \
--peerHostName proxy-east-01.example.com \
--peerPort 389 --location west

UnboundID® Directory Proxy Server <version>
Please wait while the setup program initializes...
Reading Peer Configuration ..... Done
Applying Peer Configuration ..... Done
Updating Administration Data ..... Done
Configuring Directory Proxy Server ..... Done
Configuring Certificates ..... Done
Updating Topology Admin Data ..... Done
Starting Directory Proxy Server ..... Done

To see basic server configuration status and configuration you can
launch /bin/status

See /logs/tools/setup.log for a detailed log of this operation.
```

5. Finally, repeat steps 3 and 4 to install the last directory proxy server by first copying the keystore and pin files to the `config` directory and then running the `setup` command.

```
root@proxy-west-02: cp ../*Keystore* config/
root@proxy-west-02: cp ../*Truststore* config/
```

```
root@proxy-west-02: ./setup --no-prompt \
--listenAddress proxy-west-02.example.com \
--ldapPort 389 --enableStartTLS --ldapsPort 636 \
--useJavaKeystore config/ExampleKeystore.jks \
--keyStorePasswordFile config/ExampleKeystore.pin \
--certNickName server-cert \
--useJavaTrustStore config/ExampleTruststore.jks \
--rootUserPassword pass --acceptLicense \
--aggressiveJVMTuning --maxHeapSize 1g \
--localHostName proxy-west-02.example.com \
--peerHostName proxy-east-01.example.com \
--peerPort 389 --location west

UnboundID® Directory Proxy Server <version>
Please wait while the setup program initializes...
Reading Peer Configuration ..... Done
Applying Peer Configuration ..... Done
Updating Administration Data ..... Done
Configuring Directory Proxy Server ..... Done
Configuring Certificates ..... Done
Updating Topology Admin Data ..... Done
Starting Directory Proxy Server ..... Done

To see basic server configuration status and configuration you can
launch /bin/status

See /logs/tools/setup.log for a detailed log of this operation.
```

At this point, all proxy servers have the same Admin Data backend and have the `all-servers` group defined as their configuration-server-group in the Proxy Server Global Configuration object. When making a change to a proxy server using the `dsconfig` command-line tool or the Web console, you will have the choice to apply the changes locally only or to all proxy servers in the `all-servers` group.

Testing External Server Communications After Initial Setup

After setting up the basic deployment scenario, the communication between the directory proxy server and the LDAP external servers can be tested using a feature in the directory proxy server in combination with an LDAP search.

To Test the External Communications After Initial Setup

After initial setup, the Directory Proxy Server exposes a special search base DN for testing external server connectivity, called the `backend server pass-through` subtree view. While disabled by default, you can enable this feature using `dsconfig` in the Client Connection Policy menu. Set the value of the `backend-server-passthrough-subtree-views` property to `TRUE`.

1. Run `dsconfig` to set the `include-backend-server-passthrough-subtree-views` property to `TRUE`.

```
root@proxy-east-01: dsconfig set-client-connection-policy-prop \
--policy-name default \
--set include-backend-server-passthrough-subtree-views:true
```

Once set to true, an LDAP search against the proxy with the base DN `dc=example,dc=com,ds-backend-server=ds-east-02.example.com:389` instructs the proxy to perform the search against the `ds-east-02.example.com:389` external server with the base DN set to `dc=example,dc=com`. The value of `ds-backend-server` should be the name of the configuration object representing the external server. Depending on your naming scheme, this name may not be a `host:port` combination.

2. Run `ldapsearch` to fetch the `dc=example,dc=com` entry from the `ds-east-01.example.com` server. Perform this search on each external server to determine if external server communication has been configured correctly on the directory proxy server.

```
root@proxy-east-01: bin/ldapsearch \
--bindDN "cn=Directory Manager" \
--bindPassword password \
--baseDN "dc=example,dc=com,ds-backend-server=ds-east-01.example.com:389" \
--searchScope base --useStartTLS "(objectclass=*)"
```

3. You can also use this special subtree view to track the operations performed on each external server to help determine load balancing requirements. This LDAP search can be run with the base DN values for the `ds-east-01` and `ds-east-02` servers to track the distribution of search and bind requests over time. These statistics are reset to zero when the server restarts. The following example searches an external server's monitor entry to display operation statistics:

```
root@proxy-east-01: bin/ldapsearch \
--bindDN "cn=directory manager" \
--bindPassword password \
--baseDN "cn=monitor,ds-backend-server=ds-east-02.example.com:389" \
--searchScope sub --useStartTLS "(cn=ldap*statistics)"

dn: cn=LDAP Connection Handler 192.168.1.203 port 389
Statistics,cn=monitor,ds-backend-server=ds-east-02.example.com:389

objectClass: top
objectClass: ds-monitor-entry
objectClass: ds-ldap-statistics-monitor-entry
objectClass: extensibleObject
cn: LDAP Connection Handler 192.168.1.203 port 389
Statistics
connectionsEstablished: 3004
connectionsClosed: 2990
bytesRead: 658483
bytesWritten: 2061549
ldapMessagesRead: 17278
ldapMessagesWritten: 22611
operationsAbandoned: 0
operationsInitiated: 17278
operationsCompleted: 14241
abandonRequests: 22
addRequests: 1
addResponses: 1
bindRequests: 3006
bindResponses: 3006
compareRequests: 0
compareResponses: 0
deleteRequests: 0
deleteResponses: 0
extendedRequests: 2987
extendedResponses: 2987
modifyRequests: 1
modifyResponses: 1
modifyDNRequests: 0
modifyDNResponses: 0
searchRequests: 8271
searchResultEntries: 8370
searchResultReferences: 0
searchResultsDone: 8246
unbindRequests: 2990
```

Testing a Simulated External Server Failure

Once you have tested connectivity, run a simulated failure of a load-balanced external server to verify that the directory proxy server redirects LDAP requests appropriately. In this procedure,

we stop the ds-east-01.example.com:389 server instance and test searches through proxy-east-01.example.com.

To Test a Simulated External Server Failure

1. First, perform several searches against the proxy. Verify activity in each of the servers in the east location, ds-east-01 and ds-east-02, by looking at the access logs. Because we used the default load balancing algorithm of fewest operations, it is likely that all of the searches will go to only one of the proxies. The following simple search can be repeated as needed:

```
root@proxy-east-01: bin/ldapsearch \
--bindDN "cn=Directory Manager" \
--bindPassword password --baseDN "dc=example,dc=com" \
--searchScope base --useStartTLS "(objectclass=*)"
```

2. Next, stop the directory server instance on ds-east-01.example.com using the `stop-ds` command and immediately retry the above searches. There should be no errors or noticeable delay in processing the search.

```
root@ds-east-01: bin/stop-ds

root@proxy-east-01: bin/ldapsearch \
--bindDN "cn=Directory Manager" \
--bindPassword password --baseDN "dc=example,dc=com" \
--searchScope base --useStartTLS "(objectclass=*)"
```

3. Restart the directory server instance on ds-east-01.example.com. Check the access log to confirm that the directory proxy server started to include the ds-east-01 server in round robin load-balancing within 30 seconds. The default time is 30 seconds, though you can change this default if desired.

Expanding the Deployment

In the following example deployment, the UnboundID® Directory 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 proxy servers will use StartTLS to communicate with the directory servers in the central region.

Note: Other than the ability to add to the proxy server's truststore, the `prepare-external-server` tool does not alter the proxy server configuration in any way.

The directory proxy server itself, installed on proxy-east-01.example.com, remains in the East location. This example will reconfigure round-robin load balancing between the six directory servers in three locations:

- ds-east-01.example.com
- ds-east-02.example.com
- ds-west-01.example.com

- ds-west-02.example.com
- ds-central-01.example.com
- ds-central-02.example.com

Overview of Deployment Steps

In this deployment scenario, we will take the following steps:

- Prepare the new external servers using the `prepare-external-server` tool.
- Use the `dsconfig` tool to configure the new LDAP external servers in the central data center and reconfigure the load-balancing algorithm to take these servers into account.
- Test external server communications after the servers have been configured and test a simulated external server failure.

Preparing Two New External Servers Using the prepare-external-server Tool

First, we prepare the external directory servers, ds-central-01 and ds-central-02, by creating the proxy user account and the supporting access rules. In this example, we will connect to the ds-central-01 directory server using StartTLS. Because we are using StartTLS, we need to capture the ds-central-01 server's certificate and put it in the trust store on our directory proxy server instance.

The `prepare-external-server` tool is located in the `bin` or `bat` directory of the server root directory, UnboundID-Proxy. In this example, we run the tool on the ds-east-01 instance of the directory proxy server.

To Prepare Two New External Servers Using the prepare-external-server Tool

1. Run the `prepare-external-server` tool to prepare the two new servers. On the first attempted bind to the server, the tool will report a "failed to bind" message as it cannot bind to the `cn=Proxy User` entry due to its not being created yet. The tool sets up the `cn=Proxy User` entry so that the Directory Proxy Server can access it and tests the communication settings to the server.

```
root@proxy-east-01: ./prepare-external-server \
--hostname ds-central-01.example.com --port 389 \
--baseDN dc=example,dc=com \
--proxyBindPassword password  \
--useStartTLS \
--proxyTrustStorePath ../config/ExampleTruststore.jks

Failed to bind as ‘cn=Proxy User’

Would you like to create or modify root user ‘cn=Proxy User” so that it is
available for this 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. We repeat the process on the other new server in the central location, ds-central-02.

```
root@proxy-east-01: ./prepare-external-server \
--hostname ds-central-02.example.com --port 389 \
--baseDN dc=example,dc=com \
--proxyBindPassword password \
--useStartTLS \
--proxyTrustStorePath ../config/ExampleTruststore.jks

Failed to bind as ‘cn=Proxy User,cn=Root DNs,cn=config’

Would you like to create or jmodify root user ‘cn=Proxy User,cn=Root
DNs,cn=config’ so that it is available for this Proxy Server? (yes /
no)[yes]:

Enter the DN of an account on ds-central-02:389 with which to create or manage the
‘cn=Proxy User,cn=Root DNs,cn=config’ account [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
```

Adding the New Directory Servers to the Proxy

After preparing the external directory servers to communicate with the Directory Proxy Server, we can now add the two servers in the central location to the directory proxy server instance. Because we have run the `prepare-external-server` tool, the two servers have the `cn=Proxy User` entry configured.

To Add the New Directory Servers to the Proxy

- Run the `dsconfig` tool, which is located in the `bin` or `bat` directory of the server root directory, UnboundID-Proxy.

```
root@proxy-east-01:./dsconfig

>>>> Specify LDAP connection parameters

Directory Proxy Server hostname or IP address [localhost]:

How do you want to connect to the Directory Proxy Server at
localhost?

    1)  LDAP
    2)  LDAP with SSL
    3)  LDAP with StartTLS

Enter choice [1]: 1

Directory Proxy Server at localhost port number [389]:
Administrator user bind DN [cn=Directory Manager]:
Password for user 'cn=Directory Manager':
```

Adding New Locations

First, we add a new central location, to which our new directory servers will be added.

To Add a New Location

The following steps show how to add the new servers to a new location using `dsconfig` interactive.

1. Run `dsconfig` and enter the LDAP connection parameters when prompted.

```
$ bin/dsconfig
```

2. On the UnboundID® Directory Proxy Server configuration console main menu, enter the number corresponding to **Location**.

```
>>>> UnboundID® Directory Proxy Server configuration console main menu

What do you want to configure?

    1)  Client Connection Policy  9)   Log Publisher
    2)  Connection Handler        10)  Log Retention Policy
    3)  External Server           11)  Log Rotation Policy
    4)  Global Attribute Index    12)  Placement Algorithm
    5)  Global Configuration      13)  Proxy Transformation
    6)  LDAP Health Check         14)  Request Processor
    7)  Load-Balancing Algorithm  15)  Subtree View
    8)  Location                  16)  Work Queue

    o)  'Basic' objects are shown - change this
    q)  quit

Enter choice: 8
```

3. On the Location management menu, enter the numer corresponding to creating a new location.

```
>>>> Location management 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]: 2
```

4. Enter "n" to create a new location from scratch.

```
>>>> Choose how to create the new Location:

    n)  new Location created from scratch
    t)  use an existing Location as a template

    b)  cancel
    q)  quit

Enter choice [n]:

>>>> Enter a name for the Location that you want to create: central
```

5. Configure the `preferred-failover-location` property of the new location so that this location fails over first to the east location and then to the west location, should all of the servers in the central location become unavailable.

```
>>>> 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

>>>> Configuring the 'preferred-failover-location' property

Specifies a set of alternate Locations in which servers may be
accessed if no servers in this Location are available. If multiple
values are provided, then the order in which the Locations are
listed is the order in which they should be tried.

Do you want to modify the 'preferred-failover-location' property?

    1)  Leave undefined
    2)  Add one or more values

    ?)  help
    q)  quit

Enter choice [1]: 2
```

6. Add the east and west locations as values of the property, specifying them in the order that they will be used for failover.

```
Select the Locations you wish to add:

    1)  east
    2)  west
    3)  Create a new Location
    4)  Add all Locations

    ?)  help
    b)  back
    q)  quit

Enter one or more choices separated by commas [b]: 1,2
```

7. Confirm that these are the correct values and finish configuring the location.

```
>>>> Configuring the 'preferred-failover-location' property
(Continued)

The 'preferred-failover-location' property references the
following Locations:

    1)  east
    2)  west

Do you want to modify the 'preferred-failover-location'
property?

    1)  Use these values
    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  east, west
    ?)  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

>>>> Apply Change To Server Group or Single Server

The current server has been configured to keep its
configuration synchronized with all servers in
the 'all-servers' server group, which is composed
of servers with ids [proxy-east-01:389,proxy-east-02:389,
proxy-west-01:389,proxy-west-02:389]. When applying a
change to a group of servers, every server must be
available and validate the change before it's
applied to any server.

Do you wish to update all servers in the group or
only the currently selected server?

    g) Update all servers in the ’all-servers’ group
    s) Update only the current server

    b) back
    c) cancel

Enter choice: g

The Location was created successfully
```

Editing the Existing Locations

Next, we edit the existing east and west locations to include the new central location in their failover logic. The new failover logic will be based on geographic distance, so that the east location will first fail over to central and then the west location.

To Edit Existing Locations

The following example procedure uses `dsconfig` interactive mode to edit the east location.

1. Run `dsconfig` and enter the LDAP connection parameters when prompted.
2. On the Directory Proxy Server console configuration menu, enter the number corresponding to Location.
3. On the Location management menu, enter the number corresponding to viewing and editing an existing location. Then, enter the number corresponding to the Location that you want to edit.

```
>>>> Location management 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)  central
    2)  east
    3)  west

    b)  back
    q)  quit

Enter choice [b]: 2
```

4. Remove the west location from the `preferred-failover-location` property. We will add it back later.

```
>>>> Configuring the 'preferred-failover-location' property

Specifies a set of alternate Locations in which servers may
be accessed if no servers in this Location are available. If
multiple values are provided, then the order in which the Locations
are listed is the order in which they should be tried.

Do you want to modify the 'preferred-failover-location'
property?

    1)  Keep the value: west
    2)  Add one or more values
    3)  Remove one or more values
    4)  Leave undefined

    ?)  help
    q)  quit

Enter choice [1]: 3

Select the Locations you wish to remove:

    1) west
    ?)  help

    b)  back
    q)  quit

Enter one or more choices separated by commas [b]: 1
```

5. Add a new value to the `preferred-failover-location` property.

```
>>>> Configuring the 'preferred-failover-location' property
(Continued)

Do you want to modify the 'preferred-failover-location'
property?

    1)  Leave undefined
    2)  Add one or more values
    3)  Revert changes

    ?)  help
    q)  quit

Enter choice [1]: 2
```

6. Select the values of the new failover locations for the east. We enter 1 and then 3, so that east will failover first to the central location and then to the west location.

```
Select the Locations you wish to add:
```

```
    1)  central
    2)  east
    3)  west
    4)  Create a new Location
    5)  Add all Locations

    ?)  help
    b)  back
    q)  quit

Enter one or more choices separated by commas [b]: 1,3

>>>> Configuring the 'preferred-failover-location' property
(Continued)

The 'preferred-failover-location' property references the
following Locations:

    1)  central
    2)  west

Do you want to modify the 'preferred-failover-location'
property?

    1)  Use these values
    2)  Add one or more values
    3)  Remove one or more values
    4)  Leave undefined
    5)  Revert changes

    ?)  help
    q)  quit

Enter choice [1]:
```

7. Confirm the new configuration information and then enter `f` to save the changes.

```
>>>> Configure the properties of the Location

        Property                    Value(s)
        ----------------------------------------
     1) description                 -
     2) preferred-failover-location central, west

     ?) help
     f) finish - apply any changes to the Location
     d) display the equivalent dsconfig command lines to either
         recreate this object or only to apply pending changes

     b)  back
     q)  quit

Enter choice [b]: f

The current server has been configured to keep its
configuration synchronized with all servers in the
'all-servers' server group, which is composed of
servers with ids [proxy-east-01:389,proxy-east-02:389,
proxy-west-01:389,proxy-west-02:389]. When applying a
change to a group of servers, every server must be
available and validate the change before it's applied
to any server. Do you wish to update all servers in
the group or only the currently selected server?

     g) Update all servers in the ’all-servers’ group
     s) Update only the current server

     b) back
     c) cancel

Enter choice: g

The Location was created successfully
```

8. Repeat steps 2-7 to reconfigure the failover logic for the west location to include the new central location. On the Location management menu, select "View and edit an existing Location," and then enter the number corresponding to select the west location.

```
>>>> Location management 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)  central
    2)  east
    3)  west

    b)  back
    q)  quit

Enter choice [b]: 3
```

9. Enter the number corresponding to corresponding to the `preferred-failover-location` property.

```
>>>> Configure the properties of the Location

       Property                     Value(s)
       -----------------------------------
   1)  description                  -
   2)  preferred-failover-location  east

   ?)  help
   f)  finish - apply any changes to the Location
   d)  display the equivalent dsconfig command lines
        to either recreate this object or only to apply
        pending changes
   b)  back
   q)  quit

Enter choice [b]: 2
```

10. Remove the east location from the `preferred-failover-location` property. We will add it back later.

```
>>>> Configuring the 'preferred-failover-location' property

Specifies a set of alternate Locations in which servers may be
accessed if no servers in this Location are available. If
multiple values are provided, then the order in which the
Locations are listed is the order in which they should be tried.

Do you want to modify the 'preferred-failover-location'
property?

    1)  Keep the value: east
    2)  Add one or more values
    3)  Remove one or more values
    4)  Leave undefined

    ?)  help
    q)  quit

Enter choice [1]: 3

Select the Locations you wish to remove:
```

```
    1) east
    ?) help

    b) back
    q) quit

Enter one or more choices separated by commas [b]: 1
```

11. Add a new value to the `preferred-failover-location` property.

```
>>>> Configuring the 'preferred-failover-location' property
(Continued)

Do you want to modify the 'preferred-failover-location'
property?

    1)  Leave undefined
    2)  Add one or more values
    3)  Revert changes

    ?)  help
    q)  quit

Enter choice [1]: 2
```

12. Select the values of the new failover locations for the west. We enter the number corresponding to the central location, so that west location will fail over first to the central location and then to the east location.

```
Select the Locations you wish to add:

    1)  central
    2)  east
    3)  west
    4)  Create a new Location
    5)  Add all Locations

    ?)  help
    b)  back
    q)  quit

Enter one or more choices separated by commas [b]: 1,2

>>>> Configuring the 'preferred-failover-location' property
(Continued)

The 'preferred-failover-location' property references the
following Locations:

    1)  central
    2)  east

Do you want to modify the 'preferred-failover-location' property?

    1)  Use these values
    2)  Add one or more values
    3)  Remove one or more values
    4)  Leave undefined
    5)  Revert changes

    ?)  help
    q)  quit

Enter choice [1]:
```

13. Confirm the new configuration information and then enter `f` to save our changes.

```
>>>> Configure the properties of the Location

        Property                    Value(s)
        ----------------------------------------
     1) description                 -
```

```
     2) preferred-failover-location  central, east

     ?) help
     f) finish - apply any changes to the Location
     d) display the equivalent dsconfig command lines
         to either recreate this object or only to apply
         pending changes
     b) back
     q) quit

Enter choice [b]: f

The current server has been configured to keep its
configuration synchronized with all servers in the
'all-servers' server group, which is composed of
servers with ids [proxy-east-01:389,proxy-east-02:389,
proxy-west-01:389,proxy-west-02:389]. When applying
a change to a group of servers, every server must be
available and validate the change before it's applied
to any server. Do you wish to update all servers in
the group or only the currently selected server?

     g) Update all servers in the 'all-servers' group
     s) Update only the current server
     b) back
     c) cancel

Enter choice: g

The Location was created successfully
```

14. List the locations to confirm that the new location was added correctly.

```
>>>> Location management 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]: 1

Location : Type
---------:--------
central  : generic
east     : generic
west     : generic
```

Adding New Health Checks for the Central Servers

Next, we must add new health checks for the two new servers.

To Add New Health Checks for the Central Servers

1. Run `dsconfig` and enter the LDAP connection parameters when prompted.

```
>>>> UnboundID® Directory Proxy Server configuration console main menu

What do you want to configure?

    1)  Client Connection Policy  9)   Log Publisher
    2)  Connection Handler        10)  Log Retention Policy
    3)  External Server           11)  Log Rotation Policy
    4)  Global Attribute Index    12)  Placement Algorithm
```

```
    5)  Global Configuration      13)  Proxy Transformations
    6)  LDAP Health Check         14)  Request Processor
    7)  Load-Balancing Algorithm  15)  Subtree View
    8)  Location                  16)  Work Queue

    o)  'Basic' objects are shown - change this
    q)  quit

Enter choice: 6
```

2. Select the number corresponding to creating a new health check.

```
>>>> LDAP Health Check management menu

What would you like to do?

    1)  List existing LDAP Health Checks
    2)  Create a new LDAP Health Check
    3)  View and edit an existing LDAP Health Check
    4)  Delete an existing LDAP Health Check

    b)  back
    q)  quit

Enter choice [b]: 2
```

3. Enter `t` to use an existing health check as a template.

```
>>>> Choose how to create the new LDAP Health Check:

    n)  new LDAP Health Check created from scratch
    t)  use an existing LDAP Health Check as a template

    b)  back
    q)  quit

Enter choice [n]: t
```

4. Enter the number corresponding to the ds-east-01 health check to use it as a template for the new health check.

```
>>>> Select an existing LDAP Health Check to use as a template
for the new LDAP Health Check configuration:

    1)  Consume Admin Alerts
    2)  Get Root DSE
    3)  ds-east-01.example.com:389_dc_example_dc_com-search-
        health-check
    4)  ds-east-02.example.com:389_dc_example_dc_com-search-
        health-check
    5)  ds-west-01.example.com:389_dc_example_dc_com-search-
        health-check
    6)  ds-west-02.example.com:389_dc_example_dc_com-search-
        health-check

    b)  back
    q)  quit

Enter choice [n]: 3
```

5. Name the new health check using the same naming strategy we established for the other servers in the deployment. As this health check is for the ds-central-01 server, the name takes the following format:

```
>>>> Enter a name for the Search LDAP Health Check that you want to create:
ds-central-01.example.com:389_dc_example_dc_com-search-health-check
```

6. Review the configuration properties and then enter `f` to finish configuring the new health check and save our changes.

```
>>>> 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  1 s
    9)   minimum-local-degraded-response-time      500 ms
    10)  minimum-nonlocal-degraded-response-time   500 ms
    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

Enter choice [b]: f

The current server has been configured to keep its configuration
synchronized with all servers in the 'all-servers' server group,
which is composed of servers with ids [proxy-east-01:389,
proxy-east-02:389,proxy-west-01:389,proxy-west-02:389]. When
applying a change to a group of servers, every server must be
available and validate the change before it's applied to any server.
Do you wish to update all servers in the group or only the
currently selected server?

    g) Update all servers in the ’all-servers’ group
    s) Update only the current server
    b) back
    c) cancel

Enter choice: g

The Search LDAP Health Check was created successfully
```

7. Repeat steps 2-7 to create another new health check for the `ds-central-02` server.

Adding New External Servers

Next, we add new external servers by selecting “External Server” from the configuration console main menu.

To Add New External Servers

1. Run `dsconfig` and enter the LDAP connection parameters when prompted.

```
>>>> UnboundID-Proxy Directory Proxy Server configuration console main menu
```

```
What do you want to configure?

    1)  Client Connection Policy  9)   Log Publisher
    2)  Connection Handler        10)  Log Retention Policy
    3)  External Server           11)  Log Rotation Policy
    4)  Global Attribute Index    12)  Placement Algorithm
    5)  Global Configuration      13)  Proxy Transformations
    6)  LDAP Health Check         14)  Request Processor
    7)  Load-Balancing Algorithm  15)  Subtree View
    8)  Location                  16)  Work Queue

    o)  'Basic' objects are shown - change this
    q)  quit

Enter choice: 6
```

2. On the External Server management menu, enter the number corresponding to "Create a new External Server".

```
>>>> External Server management menu

What would you like to do?

    1)  List existing External Servers
    2)  Create a new External Server
    3)  View and edit an existing External Server
    4)  Delete an existing External Server

    b)  back
    q)  quit

Enter choice [b]: 2
```

3. Base the configuration of the new external server on the existing configuration of the ds-east-01 server, so we enter `t` to use an existing External Server as a template.

```
>>>> Choose how to create the new External Server:

    n)  new External Server created from scratch
    t)  use an existing External Server as a template

    b)  back
    q)  quit

Enter choice [n]: t
```

4. Enter the number to base the configuration of the new server on the configuration of the ds-east-01 server.

```
>>>> Select an existing External Server to use as a template for
the new External Server configuration:

    1)  ds-east-01.example.com:389
    2)  ds-east-02.example.com:389
    3)  ds-west-01.example.com:389
    4)  ds-west-02.example.com:389

    b)  back
    q)  quit

Enter choice [n]: 1
```

5. Enter a name for the new ds-central-01 server that complies with our naming strategy.

```
>>>> Enter a name for the UnboundID DS External Server that you
want to create: ds-central-01.example.com:389
```

6. Enter the value of the `server-host-name` property.

```
>>>> Configure the ’server-host-name’ propety
 >>>> via creating ’ds-central-01.example.com:389’

    The host name or IP address of the target LDAP server.
    Syntax:  STRING

Enter a value for the ’server-host-name’property: ds-central-01.example.com:389
```

7. Review and modify the configuration properties of the external server.

```
>>>> Configure the properties of the UnboundID DS External Server

        Property                Value(s)
        ------------------------------------------------------------
   1)   description             -
   2)   server-host-name        ds-east-01.example.com
   3)   server-port             389
   4)   location                east
   5)   bind-dn                 cn=Proxy User
   6)   password                ******
   7)   connection-security     starttls
   8)   authentication-method   simple
   9)   health-check            ds-east-01.example.com:389_dc_
                                example_dc_com-search-health-check
   10)  health-check-frequency  30 s
   11)  allowed-operation       abandon, add, bind, compare, delete,
                                    extended, modify, modify-dn, search
   12)  key-manager-provider    Null
   13)  trust-manager-provider  JKS

   ?)   help
   f)   finish - create the new UnboundID DS External Server
   a)   show advanced properties of the UnboundID DS External Server
   d)   display the equivalent dsconfig arguments to create this
        object
   b)   back
   q)   quit

Enter choice [b]: 2
```

8. On the External Server menu, change the `server-host-name` property to reflect the name of the `ds-central-01` server.

```
>>>> Configuring the 'server-host-name' property

    The host name or IP address of the target LDAP server.

    Syntax:  STRING

Do you want to modify the 'server-host-name' property?

    1)  Keep the value: ds-east-01.example.com
    2)  Change the value

    ?)  help
    q)  quit

Enter choice [1]: 2

Enter a value for the 'server-host-name' property: ds-central-01.example.com
```

9. On the External Server menu, change the `location` property to reflect the central location.

```
>>>> Configure the properties of the UnboundID DS External Server

        Property                Value(s)
        ------------------------------------------------------------
   1)   description             -
   2)   server-host-name        ds-east-01.example.com
   3)   server-port             389
   4)   location                east
   5)   bind-dn                 cn=Proxy User
```

```
   6)   password                ******
   7)   connection-security     starttls
   8)   authentication-method   simple
   9)   health-check            ds-east-01.example.com:389_dc_
                                example_dc_com-search-health-check
   10)  health-check-frequency  30 s
   11)  allowed-operation       abandon, add, bind, compare, delete,
                                    extended, modify, modify-dn, search
   12)  key-manager-provider    Null
   13)  trust-manager-provider  JKS

   ?)   help
   f)   finish - create the new UnboundID DS External Server
   a)   show advanced properties of the UnboundID DS External Server
   d)   display the equivalent dsconfig arguments to create this
        object
   b)   back
   q)   quit

Enter choice [b]: 2

>>>> Configuring the 'location' property

Specifies the location for the LDAP External Server.

Do you want to modify the 'location' property?

    1)  Keep the value: east
    2)  Change it to the Location: central
    3)  Change it to the Location: west
    4)  Create a new Location
    5)  Leave undefined

    ?)  help
    q)  quit

Enter choice [1]: 2
```

10. Change the `health-check` property to reflect the new health check we created for the ds-central-01 server in the previous section.

```
>>>> Configure the properties of the UnboundID DS External Server

        Property               Value(s)
        ---------------------------------------------------------------
   1)   description            -
   2)   server-host-name       ds-east-01.example.com
   3)   server-port            389
   4)   location               central
   5)   bind-dn                cn=Proxy User
   6)   password               ******
   7)   connection-security    starttls
   8)   authentication-method  simple
   9)   health-check           ds-east-01.example.com:389_dc_
                               example_dc_com-search-health-check
   10)  health-check-frequency 30 s
   11)  allowed-operation      abandon, add, bind, compare, delete,
                                   extended, modify, modify-dn, search
   12)  key-manager-provider   Null
   13)  trust-manager-provider JKS

   ?)   help
   f)   finish - create the new UnboundID DS External Server
   a)   show advanced properties of the UnboundID DS External Server
   d)   display the equivalent dsconfig arguments to create this
        object
   b)   back
   q)   quit

Enter choice [b]: 9
```

11. On the 'health-check' Property menu, enter the number to remove one or more values.

```
>>>> Configuring the 'health-check' property
```

```
Specifies the health check to use for the LDAP External Server.

Do you want to modify the 'health-check' property?

    1)  Keep the value:
        ds-east-01.example.com:389_dc_example_dc_com-search-
        health-check
    2)  Add one or more values
    3)  Remove one or more values
    4)  Leave undefined

    ?)  help
    q)  quit

Enter choice [1]: 3

Select the LDAP Health Checks you wish to remove:

    1)  ds-east-01.example.com:389_dc_example_dc_com-search-health-check

    ?)  help
    b)  back
    q)  quit

Enter one or more choices separated by commas [b]: 1
```

12. Add the health-check entered in the previous section.

```
>>>> Configuring the 'health-check' property (Continued)

Do you want to modify the 'health-check' property?

    1)  Leave undefined
    2)  Add one or more values
    3)  Revert changes

    ?)  help
    q)  quit

 Enter choice [1]: 2
```

13. Select the health check associated with the ds-central-01 server.

```
Select the LDAP Health Checks you wish to add:

    1)  Consume Admin Alerts              6)   ds-east-02.example.com:389_
                                               dc_example_dc_com-search-
                                               health-check
    2)  Get Root DSE                      7)   ds-west-01.example.com:389_
                                               dc_example_dc_com-search-
                                               health-check
    3)  ds-central-01.example.com:389_    8)   ds-west-02.example.com:389_
        dc_example_dc_com-search-health-       dc_example_dc_com-search-
        check                                  health-check
    4)  ds-central-02.example.com:389_    9)   Create a new Health Check
        dc_example_dc_com-search-health-
        check
    5)  ds-east-01.example.com:389_dc_    10)  Add all LDAP Health Checks
        example_dc_com-search-health-check

    ?)  help
    b)  back
    q)  quit

Enter one or more choices separated by commas [b]: 3
```

14. Press **Enter** to use the value associated with ds-central-01 health check.

```
>>>> Configuring the 'health-check' property (Continued)

Do you want to modify the 'health-check' property?

    1)  Use the value:
        ds-central-01.example.com:389_dc_example_dc_com-search-health-check
    2)  Add one or more values
    3)  Remove one or more values
    4)  Leave undefined
    5)  Revert changes

    ?)  help
    q)  quit

Enter choice [1]:
```

15. Review the configuration of the new external server and enter `f` to create the server.

```
>>>> Configure the properties of the UnboundID DS External Server

         Property                Value(s)
         ------------------------------------------------------------
    1)   description             -
    2)   server-host-name        ds-central-01.example.com
    3)   server-port             389
    4)   location                central
    5)   bind-dn                 cn=Proxy User
    6)   password                ********
    7)   connection-security     starttls
    8)   authentication-method   simple
    9)   health-check            ds-central-01.example.com:389_
                                 dc_example_dc_com-search-health-check

    10)  health-check-frequency  30 s
    11)  allowed-operation       abandon, add, bind, compare, delete,
                                 extended, modify, modify-dn, search
    12)  key-manager-provider    Null
    13)  trust-manager-provider  JKS

    ?)   help
    f)   finish - create the new UnboundID DS External Server
    a)   show advanced properties of the UnboundID DS External
         Server
    d)   display the equivalent dsconfig arguments to create this
         object

    b)   back
    q)   quit

Enter choice [b]: f

The current server has been configured to keep its configuration
synchronized with all servers in the 'all-servers' server group,
which is composed of servers with ids [proxy-east-01:389,
proxy-east-02:389,proxy-west-01:389,proxy-west-02:389]. When
applying a change to a group of servers, every server must be
available and validate the change before it's applied to any
server. Do you wish to update all servers in the group or only
the currently selected server?

    g) Update all servers in the ’all-servers’ group
    s) Update only the current server
    b) back
    c) cancel

Enter choice: g

The UnboundID DS External Server was created successfully
```

16. Repeat this procedure to add the new `ds-central-02` external server.

Modifying the Load Balancing Algorithm

We now need to modify the existing load-balancing algorithm to include the newly created servers, so we select “Load-Balancing Algorithm” from the configuration console main menu.

To Modify the Load-Balancing Algorithm

1. Run `dsconfig` and enter the LDAP connection parameters when prompted. On the

```
>>>> UnboundID® Directory Proxy Server configuration console main menu

What do you want to configure?

    1)  Client Connection Policy  9)   Log Publisher
    2)  Connection Handler        10)  Log Retention Policy
    3)  External Server           11)  Log Rotation Policy
    4)  Global Attribute Index    12)  Placement Algorithm
    5)  Global Configuration      13)  Proxy Transformations
    6)  LDAP Health Check         14)  Request Processor
    7)  Load-Balancing Algorithm  15)  Subtree View
    8)  Location                  16)  Work Queue

    o)  'Basic' objects are shown - change this
    q)  quit

Enter choice: 6
```

2. On the Load-Balancing Algorithm management menu, enter the number corresponding to "View and edit an existing Load-Balancing Algorithm".

```
>>>> Load-Balancing Algorithm management menu

What would you like to do?

    1) List existing Load-Balancing Algorithms
    2) Create a new Load-Balancing Algorithm
    3) View and edit an existing Load-Balancing Algorithm
    4) Delete an existing Load-Balancing Algorithm

    b)  back
    q)  quit

Enter choice [b]: 3

>>>> There is only one Load-Balancing Algorithm:
'dc_example_dc_com-round-robin'. Are you sure that this is
the correct one? (yes / no) [yes]:
```

3. Add the ds-central-01 and ds-central-02 servers to the `backend-server` configuration property of the round robin load-balancing algorithm.

```
>>>> Configure the properties of the Round Robin Load-Balancing Algorithm

         Property             Value(s)
         ----------------------------------------------------------------
     1)  description                                 -
     2)  enabled                                     true
     3)  backend-server                              ds-east-01.example.com:389,
                                                     ds-east-02.example.com:389,
                                                     ds-west-01.example.com:389,
                                                     ds-west-02.example.com:389
     4)  use-location                                true
     5)  prefer-degraded-servers-over-failover       false
     6)  maximum-allowed-local-response-time         30 s
     7)  maximum-allowed-nonlocal-response-time      30 s
```

```
     8)  maximum-retry-count                       1
     9)  initial-connections                       10
     10) max-connections                           64

     ?)  help
     f)  finish - apply any changes to the Round Robin Load Balancing
         Algorithm
     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]: 3
```

4. On the backend-server property menu, enter the number corresponding to adding one or more values.

```
>>>> Configuring the 'backend-server' property

Specifies the set of backend servers that will be
available to process forwarded requests.

The 'backend-server' property references the following
LDAP External Servers:

       *)  ds-east-01.example.com:389
       *)  ds-east-02.example.com:389
       *)  ds-west-01.example.com:389
       *)  ds-west-02.example.com:389

Do you want to modify the 'backend-server' property?

       1)  Keep these values
       2)  Add one or more values
       3)  Remove one or more values
       4)  Remove all values

       ?)  help
       q)  quit

Enter choice [1]: 2
```

5. Select the external servers to add. In this example, select ds-central-01.example.com and ds-central-02.example.com.

```
Select the LDAP External Servers you wish to add:

    1)  ds-central-01.example.com:389
    2)  ds-central-02.example.com:389
    3)  Create a new LDAP External Server
    4)  Add all LDAP External Servers

    ?)  help
    b)  back
    q)  quit

Enter one or more choices separated by commas [b]: 1,2

>>>> Configuring the 'backend-server' property (Continued)

The 'backend-server' property references the following LDAP
External Servers:

    *)  ds-central-01.example.com:389
    *)  ds-central-02.example.com:389
    *)  ds-east-01.example.com:389
    *)  ds-east-02.example.com:389
    *)  ds-west-01.example.com:389
    *)  ds-west-02.example.com:389

Do you want to modify the 'backend-server' property?

    1)  Use these values
    2)  Add one or more values
```

```
    3)  Remove one or more values
    4)  Remove all values
    5)  Revert changes

    ?)  help
    q)  quit

Enter choice [1]:
```

6. Review the changes made to the load-balancing algorithm's configuration properties. When we are satisfied, we enter `f` to save our changes.

```
>>>> Configure the properties of the Round Robin Load-Balancing Algorithm

        Property                                 Value(s)
        -----------------------------------------------------------
  1)    description                              -
  2)    enabled                                  true
  3)    backend-server                           ds-central-01.example.com:389,
                                                 ds-central-02.example.com:389,
                                                 ds-east-01.example.com:389,
                                                 ds-east-02.example.com:389,
                                                 ds-west-01.example.com:389,
                                                 ds-west-02.example.com:389
  4)   use-location                             true
  5)   prefer-degraded-servers-over-failover    false
  6)   maximum-allowed-local-response-time      30 s
  7)   maximum-allowed-nonlocal-response-time   30 s
  8)   maximum-retry-count                      1
  9)   initial-connections                      10
  10)  max-connections                          64

  ?)   help
  f)   finish - apply any changes to the Round Robin Load Balancing
       Algorithm
  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

The current server has been configured to keep its configuration synchronized
with all servers in the 'all-servers' server group, which is composed of servers
with ids [proxy-east-01:389,proxy-east-02:389,proxy-west-01:389,proxy-west-02:389].
When applying a change to a group of servers, every server must be available and
validate the change before it's applied to any server. Do you wish to update all
servers in the group or only the currently selected server?

   g) Update all servers in the 'all-servers' group
   s) Update only the current server
   b) back
   c) cancel

Enter choice: g

The Round Robin Load-Balancing Algorithm was modified successfully
```

The change has been saved and applied to the directory proxy server. The load-balancing algorithm is referenced in the load-balancing-algorithm property of the request processor used by this directory proxy server.

7. To view this property, go to the main configuration console menu and select "Request Processor".

```
>>>> UnboundID® Directory Proxy Server configuration console main menu

What do you want to configure?

    1)  Client Connection Policy  9)   Log Publisher
    2)  Connection Handler        10)  Log Retention Policy
    3)  External Server           11)  Log Rotation Policy
```

```
    4)  Global Attribute Index    12)  Placement Algorithm
    5)  Global Configuration      13)  Proxy Transformations
    6)  LDAP Health Check         14)  Request Processor
    7)  Load-Balancing Algorithm  15)  Subtree View
    8)  Location                  16)  Work Queue

    o)  'Basic' objects are shown - change this
    q)  quit

Enter choice: 14
```

8. On the Request Processor management menu, enter the number corresponding to view and edit an existing request processor.

```
>>>> Request Processor management menu

What would you like to do?

    1)  List existing Request Processors
    2)  Create a new Request Processor
    3)  View and edit an existing Request Processor
    4)  Delete an existing Request Processor

    b)  back
    q)  quit

Enter choice [b]: 3
```

9. Select the request process used by your proxy server. The configuration properties are displayed as follows:

```
>>>> Configure the properties of the Proxying Request Processor

        Property                  Value(s)
        ------------------------------------------------------------------
    1)  description               -
    2)  enabled                   true
    3)  allowed-operation         abandon, add, bind, compare, extended
                                  modify, modify-dn, search
    4)  load-balancing-algorithm  dc_example_dc_com-round-robin
    5)  transformation            -
    6)  referral-behavior         pass-through
    7)  supported-control         account-usable, assertion,
                                  authorization-identity,
                                  get-authorization-entry,
                                  get-effective-rights,
                                  ignore-no-user-modification,
                                  intermediate-client, manage-dsa-it,
                                  matched-values, no-op,
                                  password-policy,
                                  post-read, pre-read,
                                  proxied-authorization-v1,
                                  proxied-authorization-v2,
                                  real-attributes-only,
                                  retain-identity,
                                  server-side-sort, subentries,
                                  subtree delete,
                                  virtual-attributes-only
     8)  supported-control-oid    -

     ?)  help
     f)  finish - apply any changes to the Proxying Request Processor
     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]:
```

This request processor is used by the subtree view serviced by the directory proxy server, which is in turn referenced by the client connection policy.

Note: The changes made in this procedure are already in effect. The directory proxy server does not have to be restarted.

Testing External Server Communication

After adding and configuring the new external servers, test the communication between the directory proxy server and the LDAP external servers using the `include-backend-server-passthrough-subtree-views` property of the directory proxy server in combination with an LDAP search. For more information about this option, see *Testing External Server Communications* on page 190.

To Test External Server Communication

- Run the `ldapseasrch` command to test communications on the ds-central-01 serverTask.

```
root@proxy-east-01: bin/ldapsearch --port 389 --bindDN "cn=directory manager" \
--bindPassword password \
--baseDN "dc=example,dc=com,ds-backend-server=ds-central-01.example.com:389" \
--searchScope base "(objectclass=*)"
```

  You can repeat this search on the ds-central-02 server, to confirm that the server returns the entry as expected.

Testing a Simulated External Server Failure

Once you have tested connectivity, run a simulated failure of a load-balanced external server to verify that the directory proxy server redirects LDAP requests appropriately. We stop the ds-east-01.example.com:389 and ds-east-02.example.com:389 server instances and test searches through proxy-east-01.example.com.

To Test a Simulated External Server Failure

1. We stop the ds-east-01.example.com:389 and ds-east-02.example.com:389 server instances and test searches through proxy-east-01.example.com.

2. Perform several searches against the proxy. Verify activity in each of the servers in the east location, ds-east-01 and ds-east-02, by looking at the access logs. The following simple search can be repeated as needed:

```
root@proxy-east-01: bin/ldapsearch --bindDN "cn=Directory Manager" \
--bindPassword password --baseDN "dc=example,dc=com" \
--searchScope base --useStartTLS "(objectclass=*)"
```

3. Next, stop the directory server instance on ds-east-01.example.com and ds-east-02.example.com using the stop-ds command and immediately retry the above searches. There should be no errors or noticeable delay in processing the search.

```
root@proxy-east-01: bin/stop-ds
```

```
root@proxy-east-01: bin/ldapsearch \
--bindDN "cn=Directory Manager" --bindPassword password \
--baseDN "dc=example,dc=com" --searchScope base --useStartTLS \
"(objectclass=*)"
```

4. Check the access log to confirm that requests made to these servers are routed to the central servers, as these servers are the first failover location in the failover list for the ds-east-01 and ds-east-02 servers.

5. Restart the directory server instance on ds-east-01.example.com and ds-east-02.example.com. Check their access logs to ensure that traffic is redirected back from the failover servers.

Merging Two Data Sets Using Proxy Transformations

In the following example, the Example.com company acquires Sample Corporation. During the merger, Example.com migrates data from Sample's o=sample rooted directory, converting Sample's sampleAccount auxiliary object class usage to Example.com's exampleAccount object class for entries rooted under `dc=example,dc=com`. Knowing that it can take considerable time for Sample's directory clients to become aware of the new DIT and schema, proxy data transformations are created to give the Sample clients as consistent a view of the data as possible during the migratory period. These transformations allow the clients to search and modify entries under o=sample using the Sample Corp. schema.

Overview of the Attribute and DN Mapping

To achieve the merger of the two data sets, we create proxy transformations that map the Sample source attributes to Example.com target attributes as described in Table 9-1, "Attribute Mapping". The Example.com schema already defines an attribute to contain the RDN of user entries, called `uid`. However, Example.com chooses to create two new attributes within its exampleAccount object class to accommodate two attributes in the Sample schema for representing the region and the DN of linked accounts.

During the merger, Example.com decides to re-parent Sample's customer entries, which are defined under two different subtrees, ou=east,o=sample and ou=west,o=sample, placing them under Example.com's `ou=people,dc=example,dc=com` subtree. Associated proxy transformations are described in Table 9-2, "DN Mappings". In this process, Example.com collapses the Sample tree, moving entries from the east and west region under a single DN, `dc=example,dc=com`. The DN proxy transformations assume that all Sample users have been co-located under this single Example.com subtree.

Table 8: Attribute Mapping

Sample Attribute	Example.com Attribute	Description
sampleID	uid	RDN of user entries
sampleRegion	exSampleRegion	String value representing the region
sampleLinkedAccounts	exSampleLinkedAccounts	DN value

Legacy Sample LDAP applications searching for entries in either the Sample base DN `ou=east,o=sample` or `ou=west,o=sample` will be successfully serviced, though there will be one or more differences in the user entries seen by the Sample legacy applications. Since the Example.com directory has no knowledge of the Sample user's former `ou=east` or `ou=west` association, search results for client searching under `o=sample` will return a DN that may differ from the original search base. For instance, a search for `sampleID=abc123` under `ou=west,o=sample` may return the user entry for abc123 with the DN of `sampleID=abc123,ou=east,o=sample`. The following table illustrates the mapping DNs.

Table 9: DN Mapping

Sample DN	Example.com DN
ou=east,o=sample	dc=example,dc=com
ou=west,o=sample	dc=example,dc=com
o=sample	dc=example,dc=com

About Mapping Multiple Source DNs to the Same Target DN

Some complications exist when defining multiple DN mappings that are used for the same request processor and the same source or target DN (or that have source or target DNs that are hierarchically related). The client request may not include enough information to disambiguate and determine the proper rule to follow.

Several solutions exist to avoid problems of disambiguation. If the client does not need to be able to see all mappings at the same time, then a new client connection policy can be created to use connection criteria that select the set of mappings applied to the client based on information such as the IP address or bind DN. Each client connection policy would have separated subtree views with separate proxying request processors that reference the appropriate transformation for that client.

Alternatively, if it is unnecessary to search under the o=sample base DN, then separate subtree views can be created in the same client connection policy. For example, one subtree view would be created for `ou=east,o=sample` and one for `ou=west,o=sample`. Each subtree view is then associated with its own proxying request processor, one for ou=east requests and one for ou=west requests.

An Example of a Migrated Sample Customer Entry

The following example is an example of a Sample customer entry that has been migrated to the Example.com database. The user entry is defined in the Example.com directory server's database as follows. The attributes that have undergone a proxy transformation are marked in bold. Note that this view is how the entry appears to search requests under the `dc=example,dc=com` base DN.

```
dn: uid=scase,ou=People,dc=example,dc=com
objectClass: person
objectClass: inetOrgPerson
objectClass: organizationalPerson
objectClass: exampleAccount
objectClass: top
description: A customer account migrated from Sample merger
uid: scase
exAccountNumber: 234098
```

```
exSampleRegion: east
exSampleLinkedAccounts: uid=jcase,ou=people,dc=example,dc=com
userPassword: password
givenName: Sterling
cn: Sterling Case
sn: Case
telephoneNumber: +1 804 094 3356
street: 00468 Second Street
l: Arlington
mail: sterlingcase@maildomain.com st: VA
```

The following examples shows what the directory proxy server returns to LDAP clients who have requested the entry when searching under the o=sample base DN. Note that the DN returned includes ou=east, even though this branch does not exist in the Example.com DIT. It also returns the attribute names as they are defined in the Sample schema.

```
dn:  sampleID=scase,ou=east,o=sample
objectClass: person
objectClass: inetOrgPerson
objectClass: organizationalPerson
objectClass: exampleAccount
objectClass: top
description: A customer account migrated from Sample merger
uid: scase
exAccountNumber: 234098
exSampleRegion: east
exSampleLinkedAccounts: sampleID=jcase,ou=people,dc=example,dc=com
userPassword: password
givenName: Sterling
cn: Sterling Case
sn: Case
telephoneNumber: +1 804 094 3356
street: 00468 Second Street
l: Arlington
mail: sterlingcase@maildomain.com st: VA
```

Overview of Deployment Steps

In this deployment scenario, we will take the following steps:

- Install any necessary schema on the directory proxy server.
- Create three attribute mapping proxy transformations and three DN mapping proxy transformations
- Create a new proxying request processor, using the existing dc_example_dc_com request processor as a template.
- Assign the six proxy transformations to the new proxying request processor.
- Create a new subtree view for o=sample that references the new proxying request processor.
- Add the new subtree view to the existing client connection policy.
- Test our configuration by performing some searches on the Sample DIT.

About the Schema

The directory proxy server inherits user-defined schema from all external servers by comparing `cn=schema` on these servers at proxy server startup and at five minute intervals. As a result, example.com schema does not need to be added manually to the proxy server's `config/schema`

directory. We assume that the schema for Sample entries has been defined on the external servers with the example.com DIT, requiring no direct schema management on the proxy server. The following schema definitions are assumed to exist on the external directory server:

```
dn: cn=schema
objectClass: top
objectClass: ldapSubentry
objectClass: subschema
cn: schema
attributeTypes: ( 1.3.6.1.4.1.32473.2.1.1
  NAME 'exAccountNumber'
  SYNTAX 1.3.6.1.4.1.1466.115.121.1.15
  SINGLE-VALUE )
attributeTypes: ( 1.3.6.1.4.1.32473.1.1.3
  NAME 'sampleLinkedAccounts'
  SYNTAX 1.3.6.1.4.1.1466.115.121.1.12 )
attributeTypes: ( 1.3.6.1.4.1.32473.1.1.2
  NAME 'sampleRegion'
  SYNTAX 1.3.6.1.4.1.1466.115.121.1.15
  SINGLE-VALUE )
attributeTypes: ( 1.3.6.1.4.1.32473.1.1.1
  NAME 'sampleID'
  SYNTAX 1.3.6.1.4.1.1466.115.121.1.15
  SINGLE-VALUE )
attributeTypes: ( 1.3.6.1.4.1.32473.2.1.3
  NAME 'exSampleLinkedAccounts'
  SYNTAX 1.3.6.1.4.1.1466.115.121.1.12 )
attributeTypes: ( 1.3.6.1.4.1.32473.2.1.2
  NAME 'exSampleRegion'
  SYNTAX 1.3.6.1.4.1.1466.115.121.1.15
  SINGLE-VALUE )
objectClasses: ( 1.3.6.1.4.1.32473.2.2.1
  NAME 'exampleAccount'
  SUP top
  AUXILIARY
  MAY ( exAccountNumber $
        exSampleRegion $
        exSampleLinkedAccounts $
        sampleID $
        sampleRegion $
        sampleLinkedAccounts ) )
```

The schema file defines some Example.com schema, such as `exAccountNumber` and `exSampleRegion`, and some Sample schema, such as `sampleRegion` and `sampleID`.

Creating Proxy Transformations

We create three attribute mapping proxy transformations and three DN mapping proxy transformations. We run the `dsconfig` tool, which is located in the bin or bat directory of the server root directory, UnboundID-Proxy.

To Create Proxy Transformations

1. In the main installation directory, UnboundID-Proxy, we run the `start-proxy` command.

   ```
   root@proxy-east-01: bin/start-proxy
   ```

2. Run `dsconfig` in interactive mode and enter the LDAP connection parameters.

   ```
   root@proxy-east-01:./dsconfig

   >>>> Specify LDAP connection parameters

   Directory Proxy Server hostname or IP address [localhost]:
   ```

```
How do you want to connect to the Directory Proxy Server at localhost?

    1)  LDAP
    2)  LDAP with SSL
    3)  LDAP with StartTLS

Enter choice [1]: 1

Directory Proxy Server at localhost port number [389]:
Administrator user bind DN [cn=Directory Manager]:
Password for user 'cn=Directory Manager':
```

3. On the UnboundID® Directory Proxy Server configuration console main menu, enter the number corresponding to **Proxy Transformation**.

```
>>>> UnboundID® Directory Proxy Server configuration console main menu

What do you want to configure?

    1)  Client Connection Policy  9)   Log Publisher
    2)  Connection Handler        10)  Log Retention Policy
    3)  External Server           11)  Log Rotation Policy
    4)  Global Attribute Index    12)  Placement Algorithm
    5)  Global Configuration      13)  Proxy Transformation
    6)  LDAP Health Check         14)  Request Processor
    7)  Load-Balancing Algorithm  15)  Subtree View
    8)  Location                  16)  Work Queue

    o)  'Basic' objects are shown - change this
    q)  quit

Enter choice: 13
```

Creating the Attribute Mapping Proxy Transformations

Next, we create the attribute mapping proxy transformations using dsconfig interactive. We assume for this example that we are continuing from the previous `dsconfig` session. In the following example, this transformation maps `ou=east,o=sample` in the Sample schema `dc=exam-ple,dc=com` in the Example.com schema.

To Creating the Attribute Mapping Proxy Transformations

1. On the Proxy Transformation management menu, enter the number corresponding to "Create a New Proxy Transformation".

```
>>>> Proxy Transformation management menu

What would you like to do?

    1)  List existing Proxy Transformations
    2)  Create a new Proxy Transformation
    3)  View and edit an existing Proxy Transformation
    4)  Delete an existing Proxy Transformation

    b)  back
    q)  quit

Enter choice [b]: 2
```

2. Create a mapping from the sampleRegion attribute to the `exSampleRegion` attribute, enter the number corresponding to "Attribute Mapping Proxy Transformation".

```
>>>> Select the type of Proxy Transformation that you want to create:
```

```
    1)  Attribute Mapping Proxy Transformation
    2)  Default Value Proxy Transformation
    3)  DN Mapping Proxy Transformation
    4)  Groovy Scripted Proxy Transformation
    5)  Simple To External Bind Proxy Transformation
    6)  Suppress Attribute Proxy Transformation
    7)  Suppress Entry Proxy Transformation
    8)  Third Party Proxy Transformation

    ?)  help
    c)  cancel
    q)  quit

Enter choice [c]: 1
```

3. Enter a descriptive name for the new proxy transformation that illustrates the attribute mapping that it performs.

```
>>>> Enter a name for the Attribute Mapping Proxy
Transformation that you want to create: sampleRegion-to-exSampleRegion
```

4. Press **Enter** to enable the proxy transformation.

```
>>>> Configuring the 'enabled' property

Indicates whether this proxy transformation 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
```

5. Provide the name of the source attribute in the Sample schema that we are mapping to the Example.com schema, which is sampleRegion.

```
>>>> Configuring the 'source-attribute' property

    Specifies the name of the attribute that may appear in
    client requests which should be remapped to the target
    attribute. Note that the source attribute must not be
    equal to the target attribute.

    Syntax:  STRING

Enter a value for the 'source-attribute' property:
sampleRegion
```

6. Review the configuration properties, and then enter `f` to create the new attribute mapping proxy transformation.

```
>>>> Configure the properties of the Attribute Mapping Proxy Transformation

        Property               Value(s)
        ----------------------------------------------------------
    1)  description            -
    2)  enabled                true
    3)  source-attribute       sampleRegion
    4)  target-attribute       exSampleRegion
    5)  enable-dn-mapping      true
    6)  enable-control-mapping true
    7)  map-control            assertion-request,
                               authorization-identity-response,
                               entry-change-notification,
```

```
                                get-authorizaton-entry-request,
                                get-authorizaton-entry-response,
                                interactive-transaction-
                                specification-response,
                                intermediate-client-request,
                                matched-values-request,
                                post-read-request,
                                post-read-response,
                                pre-read-request,
                                pre-read-response,
                                proxied-authorization-v1-request,
                                proxied-authorization-v2-request,
                                sort-request,sort-response
    ?)  help
    f)  finish - create the new Attribute Mapping Proxy
        Transformation
    d)  display the equivalent dsconfig arguments to create this
        object
    b)  back
    q)  quit

Enter choice [b]: f

The Attribute Mapping Proxy Transformation was created successfully
```

7. Repeat the previous steps to create another attribute mapping proxy transformation, this time to map between the Sample Corporation's sampleID attribute and the Example.com uid attribute.

```
>>>> Proxy Transformation management menu

What would you like to do?

    1)  List existing Proxy Transformations
    2)  Create a new Proxy Transformation
    3)  View and edit an existing Proxy Transformation
    4)  Delete an existing Proxy Transformation

    b)  back
    q)  quit

Enter choice [b]: 2
```

8. Use the proxy transformation that we created in the previous steps as a template for the new proxy transformation.

```
>>>> Choose how to create the new Proxy Transformation:

    n) new Proxy Transformation created from scratch
    t) use an existing Proxy Transformation as a template
    b) back
    q) quit

Enter choice [n]: t

>>>> Select an existing Proxy Transformation to use as a
template for the new Proxy Transformation configuration:

    1) sampleRegion-to-exSampleRegion
    n) new Proxy Transformation created from scratch
    c) cancel
    q) quit

Enter choice [n]: 1
```

9. Name the new attribute mapping transformation `sampleID-to-uid`.

```
>>>> Enter a name for the Attribute Mapping Proxy
Transformation that you want to create: sampleID-to-uid
```

10. Enter the new name of the source attribute.

```
>>>> Enter a value for the 'source-attribute' property: sampleID
```

11. Enter the new name of the target attribute.

```
>>>> Enter a value for the 'target-attribute' property: uid
```

12. Review the properties of the new attribute mapping proxy transformation and then enter `f` to save our changes.

```
>>>> Configure the properties of the Attribute Mapping Proxy Transformation

        Property                Value(s)
        ----------------------------------------------------------
    1)  description             -
    2)  enabled                 true
    3)  source-attribute        sampleID
    4)  target-attribute        uid
    5)  enable-dn-mapping       true
    6)  enable-control-mapping  true
    7)  map-control             assertion-request,
                                authorization-identity-response,
                                entry-change-notification,
                                get-authorization-entry-request,
                                get-authorization-entry-response,
                                interactive-transaction-
                                specification-response,
                                intermediate-client-request,
                                matched-values-request,
                                post-read-request,
                                post-read-response,
                                pre-read-request,
                                pre-read-response,
                                proxied-authorization-v1-request,
                                proxied-authorization-v2-request,
                                sort-request,sort-response
    ?)  help
    f)  finish - create the new Attribute Mapping Proxy
        Transformation
    d)  display the equivalent dsconfig arguments to create this
        object
    b)  back
    q)  quit

Enter choice [b]: f

The current server has been configured to keep its configuration
synchronized with all servers in the 'all-servers' server group,
which is composed of servers with ids [proxy-east-01:389,
proxy-east-02:389]. When applying a change to a group of servers,
every server must be available and validate the change before
it's applied to any server. Do you wish to update all servers in
the group or only the currently selected server?

    g) Update all servers in the 'all-servers' group
    s) Update only the current server
    b) back
    c) cancel

Enter choice: g

The Attribute Mapping Proxy Transformation was created successfully
```

13. Repeat the previous steps again to create a last attribute mapping proxy transformation, mapping between the Sample sampleLinkedAccounts attribute and the Example.com exSampleLinkedAccounts attribute.

```
>>>> Proxy Transformation management menu

What would you like to do?

    1) List existing Proxy Transformations
```

```
    2) Create a new Proxy Transformation
    3) View and edit an existing Proxy Transformation
    4) Delete an existing Proxy Transformation

    b)  back
    q)  quit

Enter choice [b]: 2
```

14. Use the proxy transformation that we created in the previous steps as a template for the new proxy transformation.

```
>>>> Choose how to create the new Proxy Transformation:

    n) new Proxy Transformation created from scratch
    t) use an existing Proxy Transformation as a template
    b) back
    q) quit

Enter choice [n]: t

>>>> Select an existing Proxy Transformation to use as a
template for the new Proxy Transformation configuration:

    1) sampleRegion-to-exSampleRegion
    n) new Proxy Transformation created from scratch
    c) cancel
    q) quit

Enter choice [n]: 1
```

15. Name the new attribute mapping transformation sampleLinkedAccounts-to-exSampleLinkedAccounts.

```
>>>> Enter a name for the Attribute Mapping Proxy Transformation
that you want to create: sampleLinkedAccounts-to-exSampleLinkedAccounts
```

16. Enter the new source-attribute name, `sampleLinkedAccounts`.

```
>>>> Enter a value for the 'source-attribute' property: sampleLinkedAccounts
```

17. Enter the new name of the target attribute, `exSampleLinkedAccounts`.

```
>>>> Enter a value for the 'target-attribute' property: exSampleLinkedAccounts
```

18. Review the properties of the new attribute mapping proxy transformation and then enter `f` to save our changes. Then, apply the change to all servers in the server group.

```
>>>> Configure the properties of the Attribute Mapping Proxy Transformation

        Property                Value(s)
        ------------------------------------------------------------
    1)  description             -
    2)  enabled                 true
    3)  source-attribute        sampleLinkedAccounts
    4)  target-attribute        exSampleLinkedAccounts
    5)  enable-dn-mapping       true
    6)  enable-control-mapping  true
    7)  map-control             assertion-request,
                                authorization-identity-response,
                                entry-change-notification,
                                get-authorization-entry-request,
                                get-authorization-entry-response,
                                interactive-transaction-
                                specification-response,
                                intermediate-client-request,
                                matched-values-request,
                                post-read-request,
                                post-read-response,
```

```
                                pre-read-request,
                                pre-read-response,
                                proxied-authorization-v1-request,
                                proxied-authorization-v2-request,
                                sort-request,sort-response
    ?)  help
    f)  finish - create the new Attribute Mapping Proxy
        Transformation
    d)  display the equivalent dsconfig arguments to create this
        object
    b)  back
    q)  quit

Enter choice [b]: f

The current server has been configured to keep its configuration
synchronized with all servers in the 'all-servers' server group,
which is composed of servers with ids [proxy-east-01:389,
proxy-east-02:389]. When applying a change to a group of servers,
every server must be available and validate the change before
it's applied to any server. Do you wish to update all servers in
the group or only the currently selected server?

    g) Update all servers in the ’all-servers’ group
    s) Update only the current server
    b) back
    c) cancel

Enter choice: g

The Attribute Mapping Proxy Transformation was created successfully
```

Creating the DN Mapping Proxy Transformations

Now we create the DN mapping proxy transformations.

To Create the DN Mapping Proxy Transformations

1. On the Proxy Transformation management menu, enter the number corresponding to "Create a new Proxy Transformation".

```
>>>> Proxy Transformation management menu

What would you like to do?

    1)  List existing Proxy Transformations
    2)  Create a new Proxy Transformation
    3)  View and edit an existing Proxy Transformation
    4)  Delete an existing Proxy Transformation

    b)  back
    q)  quit

Enter choice [b]: 2
```

2. Enter n to create a new Proxy Transformation from scratch.

```
>>>> Choose how to create the new Proxy Transformation:

    n) new Proxy Transformation created from scratch
    t) use an existing Proxy Transformation as a template
    b) back
    q) quit

Enter choice [n]: t
```

3. Enter the number corresponding to "DN Mapping Proxy Transformation".

```
>>>> Select the type of Proxy Transformation that you want to
create:

    1)  Attribute Mapping Proxy Transformation
    2)  Default Value Proxy Transformation
    3)  DN Mapping Proxy Transformation
    4)  Groovy Scripted Proxy Transformation
    5)  Simple To External Bind Proxy Transformation
    6)  Suppress Attribute Proxy Transformation
    7)  Suppress Entry Proxy Transformation
    8)  Third Party Proxy Transformation

    ?)  help
    c)  cancel
    q)  quit

Enter choice [c]: 3
```

4. Enter a name for the DN Mapping Proxy Transformation. This transformation maps `ou=east,o=sample` in the Sample schema `dc=example,dc=com` in the Example.com schema.

```
>>>> Enter a name for the DN Mapping Proxy Transformation that
you want to create: sample_east-to-example
```

5. Select `TRUE` to enable the transformation by default.

```
>>>> Configuring the 'enabled' property

Indicates whether this proxy transformation 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
```

6. Specify the source DN as it appears in client requests.

```
>>>> Configuring the 'source-dn' property

     Specifies the source DN that may appear in client
     requests which should be remapped to the target DN.
     Note that the source DN must not be equal to the target DN.

     Syntax: DN

Enter a value for the 'source-dn' property:
ou=east,o=sample
```

7. Specify the target DN, where requests for the source DN should be routed.

```
>>>> Configuring the 'target-dn' property

    Specifies the DN to which the source DN should be mapped.
    Note that the target DN must not be equal to the source
    DN.

    Syntax: DN

Enter a value for the 'target-dn' property: dc=example,dc=com
```

8. Review the configuration properties, and then enter f to create the new DN mapping proxy transformation.

```
>>>> Configure the properties of the DN Mapping Proxy Transformation

        Property                 Value(s)
        -----------------------------------------------------------
    1)  description              -
    2)  enabled                  true
    3)  source-dn                "ou=east,o=sample"
    4)  target-dn                "dc=example,dc=com"
    5)  enable-attribute-mapping true
    6)  map attribute            If no specific map attributes are
                                 defined but attribute mapping is
                                 enabled, then all attributes with a
                                 distinguished name or name
                                 and optional UID syntax will be
                                 examined to determine if any
                                 mapping is required.
    7)  enable-control-mapping   true
    8)  map-control              assertion-request,
                                 authorization-identity-response,
                                 entry-change-notification,
                                 get-authorizaton-entry-request,
                                 get-authorizaton-entry-response,
                                 interactive-transaction-
                                 specification-response,
                                 intermediate-client-request,
                                 matched-values-request,
                                 post-read-request,
                                 post-read-response,
                                 pre-read-request,
                                 pre-read-response,
                                 proxied-authorization-v1-request,
                                 proxied-authorization-v2-request,
                                 sort-request,sort-response
    ?)  help
    f)  finish - create the new DN Mapping Proxy Transformation
    d)  display the equivalent dsconfig arguments to create this
        object
    b)  back
    q)  quit

Enter choice [b]: f

The current server has been configured to keep its configuration
synchronized with all servers in the 'all-servers' server group,
which is composed of servers with ids [proxy-east-01:389,
proxy-east-02:389]. When applying a change to a group of servers,
every server must be available and validate the change before it's
applied to any server. Do you wish to update all servers in the
group or only the currently selected server?

    g) Update all servers in the ’all-servers’ group
    s) Update only the current server
    b) back
    c) cancel

Enter choice: g

The DN Mapping Proxy Transformation was created successfully
```

9. Create a new DN mapping proxy transformation that maps `ou=west,o=sample` in the Sample schema to `dc=example,dc=com` in the Example.com schema. We name it `sample_west-to-example`.

```
>>>> Proxy Transformation management menu

What would you like to do?

    1)  List existing Proxy Transformations
    2)  Create a new Proxy Transformation
    3)  View and edit an existing Proxy Transformation
    4)  Delete an existing Proxy Transformation
```

```
    b)  back
    q)  quit

Enter choice [b]: 2
```

10. Enter `t` to base our new proxy transformation on the one created previously.

```
>>>> Choose how to create the new Proxy Transformation:

    n)  new Proxy Transformation created from scratch
    t)  use an existing Proxy Transformation as a template
    b)  back
    q)  quit

Enter choice [n]: t
```

11. Select the `sample_east-to-example` DN mapping proxy transformation.

```
>>>> Select an existing Proxy Transformation to use as a
template for the new Proxy Transformation configuration:

    1)  sampleID-to-uid
    2)  sampleLinkedAccounts-to-exSampleLinkedAccounts
    3)  sampleRegion-to-exSampleRegion
    4)  sample_east-to-example

    n)  new Proxy Transformation created from scratch
    c)  cancel
    q)  quit

Enter choice [n]: 4
```

12. Enter a name for the new proxy tranformation `sample_west-to-example`.

```
>>>> Enter a name for the DN Mapping Proxy Transformation that
you want to create: sample_west-to-example
```

13. Enter the new source DN, `ou=west,o=sample`.

```
>>>> Enter a value for the 'source-dn' property: ou=west,o=sample
```

14. Enter the same `target-dn` value as in the previous exmple.

```
>>>> Enter a value for the 'target-dn' property: dc=example,dc=com
```

15. Review the configuration properties and enter `f` to save our changes.

```
>>>> Configure the properties of the DN Mapping Proxy Transformation

        Property                Value(s)
        ----------------------------------------------------------
    1)  description             -
    2)  enabled                 true
    3)  source-dn               "ou=west,o=sample"
    4)  target-dn               "dc=example,dc=com"
    5)  enable-attribute-mapping true
    6)  map attribute           If no specific map attributes are
                                defined but attribute mapping is
                                enabled, then all attributes with a
                                distinguished name or name
                                and optional UID syntax will be
                                examined to determine if any
                                mapping is required.
    7)  enable-control-mapping  true
    8)  map-control             assertion-request,
                                authorization-identity-response,
                                entry-change-notification,
```

```
                                  get-authorizaton-entry-request,
                                  get-authorizaton-entry-response,
                                  interactive-transaction-
                                  specification-response,
                                  intermediate-client-request,
                                  matched-values-request,
                                  post-read-request,
                                  post-read-response,
                                  pre-read-request,
                                  pre-read-response,
                                  proxied-authorization-v1-request,
                                  proxied-authorization-v2-request,
                                  sort-request,sort-response
    ?)  help
    f)  finish - create the new DN Mapping Proxy Transformation
    d)  display the equivalent dsconfig arguments to create this
        object
    b)  back
    q)  quit

Enter choice [b]: f

The current server has been configured to keep its configuration
synchronized with all servers in the 'all-servers' server group,
which is composed of servers with ids [proxy-east-01:389,
proxy-east-02:389]. When applying a change to a group of servers,
every server must be available and validate the change before it's
applied to any server. Do you wish to update all servers in the
group or only the currently selected server?

    g) Update all servers in the ’all-servers’ group
    s) Update only the current server
    b) back
    c) cancel

Enter choice: g

The DN Mapping Proxy Transformation was created successfully
```

16. Finally, create a DN mapping proxy transformation for the base DN of the Sample database.

```
>>>> Proxy Transformation management menu

What would you like to do?

    1)  List existing Proxy Transformations
    2)  Create a new Proxy Transformation
    3)  View and edit an existing Proxy Transformation
    4)  Delete an existing Proxy Transformation

    b)  back
    q)  quit

Enter choice [b]: 2
```

17. Enter `n` to create a new Proxy Transformation from scratch.

```
>>>> Choose how to create the new Proxy Transformation:

    n) new Proxy Transformation created from scratch
    t) use an existing Proxy Transformation as a template
    b) back
    q) quit

Enter choice [n]: t
```

18. Select the `sample_east-to-example` DN mapping proxy transformation.

```
>>>> Select an existing Proxy Transformation to use as a
template for the new Proxy Transformation configuration:

    1)  sampleID-to-uid
    2)  sampleLinkedAccounts-to-exSampleLinkedAccounts
    3)  sampleRegion-to-exSampleRegion
```

```
    4)  sample_east-to-example
    5)  sample_west-to-example

    n)  new Proxy Transformation created from scratch
    c)  cancel
    q)  quit

Enter choice [n]: 4
```

19. Because this proxy transformation maps `o=sample` to `dc=example,dc=com`, we name it `sample-to-example`.

```
>>>> Enter a name for the DN Mapping Proxy Transformation that
you want to create: sample-to-example
```

20. Enter the new source DN to the Sample base DN, `o=sample`.

```
>>>> Enter a value for the 'source-dn' property: o=sample
```

21. Enter the same `target-dn` value as in the previous exmple.

```
>>>> Enter a value for the 'target-dn' property: dc=example,dc=com
```

22. Review the configuration properties and enter `f` to save our changes.

```
>>>> Configure the properties of the DN Mapping Proxy Transformation

        Property                Value(s)
        ------------------------------------------------------------
    1)  description             -
    2)  enabled                 true
    3)  source-dn               "o=sample"
    4)  target-dn               "dc=example,dc=com"
    5)  enable-attribute-mapping true
    6)  map attribute           If no specific map attributes are
                                defined but attribute mapping is
                                enabled, then all attributes with a
                                distinguished name or name
                                and optional UID syntax will be
                                examined to determine if any
                                mapping is required.
    7)  enable-control-mapping  true
    8)  map-control             assertion-request,
                                authorization-identity-response,
                                entry-change-notification,
                                get-authorizaton-entry-request,
                                get-authorizaton-entry-response,
                                interactive-transaction-
                                specification-response,
                                intermediate-client-request,
                                matched-values-request,
                                post-read-request,
                                post-read-response,
                                pre-read-request,
                                pre-read-response,
                                proxied-authorization-v1-request,
                                proxied-authorization-v2-request,
                                sort-request,sort-response
    ?)  help
    f)  finish - create the new DN Mapping Proxy Transformation
    d)  display the equivalent dsconfig arguments to create this
        object
    b)  back
    q)  quit

Enter choice [b]: f

The current server has been configured to keep its configuration
synchronized with all servers in the 'all-servers' server group,
which is composed of servers with ids [proxy-east-01:389,
proxy-east-02:389]. When applying a change to a group of servers,
```

```
every server must be available and validate the change before it's
applied to any server. Do you wish to update all servers in the
group or only the currently selected server?

    g) Update all servers in the 'all-servers' group
    s) Update only the current server
    b) back
    c) cancel

Enter choice: g

The DN Mapping Proxy Transformation was created successfully
```

Creating a Request Processor to Manage the Proxy Transformations

Next, we need to create a new proxying request processor that includes our new attribute and DN mapping proxy transformations. We will use the existing `dc_example_dc_com` request processor as a template.

To Create a Request Processor to Manage Proxy Transformations

1. On the UnboundID® Directory Proxy Server configuration console main menu, enter the number corresponding to **Request Processor**.

```
>>>> UnboundID® Directory Proxy Server configuration console main menu

What do you want to configure?

    1)  Client Connection Policy  9)   Log Publisher
    2)  Connection Handler        10)  Log Retention Policy
    3)  External Server           11)  Log Rotation Policy
    4)  Global Attribute Index    12)  Placement Algorithm
    5)  Global Configuration      13)  Proxy Transformation
    6)  LDAP Health Check         14)  Request Processor
    7)  Load-Balancing Algorithm  15)  Subtree View
    8)  Location                  16)  Work Queue

    o)  'Basic' objects are shown - change this
    q)  quit

Enter choice: 14
```

2. On the Request Processor management menu, enter the number corresponding to "Create a new Request Processor".

```
>>>> Request Processor management menu

What would you like to do?

    1)  List existing Request Processors
    2)  Create a new Request Processor
    3)  View and edit an existing Request Processor
    4)  Delete an existing Request Processor

    b)  back
    q)  quit

Enter choice [b]: 2
```

3. Enter `t` to use our current request processor as a template.

```
>>>> Choose how to create the new Proxy Transformation:

    n)  new Proxy Transformation created from scratch
    t)  use an existing Proxy Transformation as a template
```

```
    b)  back
    q)  quit

Enter choice [n]: t
```

4. Press **Enter** to use our current request processor as a template.

```
>>>> Select an existing Request Processor to use as a
template for the new Request Processor configuration:

    1)  dc_example_dc_com-req-processor
    n)  new Request Processor created from scratch
    c)  cancel
    q)  quit

Enter choice [n]: 1
```

5. Provide a name for the new proxying request processor, `o_sample-req-processor`.

```
>>>> Enter a name for the Proxying Request Processor that you
want to create: o_sample-req-processor
```

6. Review the properties. The load-balancing algorithm is the same as for the previous request processor, though the transformation property must be changed. We enter the number corresponding to the `Transformation` property.

```
>>>> Configure the properties of the Proxying Request Processor

        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-round-robin
    5)  transformation            -
    6)  referral-behavior         pass-through
    7)  supported-control         account-usable, assertion,
                                  authorization-identity,
                                  get-authorization-entry,
                                  get-effective-rights,
                                  ignore-no-user-modification,
                                  intermediate-client,
                                  manage-dsa-it, matched-values,
                                  no-op, password-policy,
                                  post-read, pre-read,
                                  proxied-authorization-v1,
                                  proxied-authorization-v2,
                                  real-attributes-only,
                                  retain-identity,
                                  server-side-sort, 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

Enter choice [b]: 5
```

7. Enter the number corresponding to specify the proxy transformations that we created in the previous sections.

```
>>>> Configuring the 'transformation' property

    Specifies the types of transformations that should be
```

```
    applied to requests and responses processed by
    this Proxying Request Processor. If multiple transformations
    are provided, then they will be invoked in the specified
    order for request transformations, and in the reverse order
    for response transformations.

Do you want to modify the 'transformation' property?

    1)  Leave undefined
    2)  Add one or more values

    ?)  help
    q)  quit

Enter choice [1]: 2
```

8. Select the attribute mapping proxy transformations first, in our case 4, 5, and 6. Next, we select the DN mapping proxy transformations. The order in which we select the DN transformations is important because we have related DNs. We begin with the DNs that are lower in the tree first, and finish with the base DN transformation.

```
Select the Proxy Transformations you wish to add:

    1)  sample-to-example       5)  sampleLinkedAccounts-to-
                                    exSampleLinkedAccounts
    2)  sample_east-to-example  6)  sampleRegion-to-
                                    exSampleRegion
    3)  sample_west-to-example  7)  Create a new Proxy
                                    Transformation
    4)  sampleID-to-uid         8)  Add all Proxy Transformations

    ?)  help
    b)  back
    q)  quit

Enter one or more choices separated by commas [b]: 4,5,6,2,3,1
```

9. Confirm that the proxy transformations are listed in the correct order and press **Enter** to accept and use the values.

```
>>>> Configuring the 'transformation' property (Continued)

The 'transformation' property references the following Proxy
Transformations:

    1) sampleID-to-uid
    2) sampleLinkedAccounts-to-exSampleLinkedAccounts
    3) sampleRegion-to-exSampleRegion
    4) sample_east-to-example
    5) sample_west-to-example
    6) sample-to-example

Do you want to modify the 'transformation' property?

    1)  Use these values
    2)  Add one or more values
    3)  Remove one or more values
    4)  Leave undefined
    5)  Revert changes

    ?)  help
    q)  quit

Enter choice [1]:
```

10. Reviewing the request processor properties, we enter `f` to save our changes.

```
>>>> Configure the properties of the Proxying Request Processor

        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-round-robin
    5)  transformation             sampleID-to-uid,
                                   sampleLinkedAccounts-to-
                                   exSampleLinkedAccounts,
                                   sampleRegion-to-exSampleRegion,
                                   sample_east-to-example,
                                   sample_west-to-example,
                                   sample-to-example
    6)  referral-behavior          pass-through
    7)  supported-control          account-usable, assertion,
                                   authorization-identity,
                                   get-authorization-entry,
                                   get-effective-rights,
                                   ignore-no-user-modification,
                                   intermediate-client,
                                   manage-dsa-it, matched-values,
                                   no-op, password-policy,
                                   post-read, pre-read,
                                   proxied-authorization-v1,
                                   proxied-authorization-v2,
                                   real-attributes-only,
                                   retain-identity,
                                   server-side-sort, 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

Enter choice [b]: f

The Proxying Request Processor was created successfully
```

Creating Subtree Views

At this stage, we need to configure subtree views for the Directory Proxy Server.

To Create Subtree Views

1. On the UnboundID® Directory Proxy Server configuration console main menu, enter the number corresponding to **Subtree View**.

```
>>>> UnboundID® Directory Proxy Server configuration console main menu

What do you want to configure?

    1)  Client Connection Policy  9)   Log Publisher
    2)  Connection Handler        10)  Log Retention Policy
    3)  External Server           11)  Log Rotation Policy
    4)  Global Attribute Index    12)  Placement Algorithm
    5)  Global Configuration      13)  Proxy Transformation
    6)  LDAP Health Check         14)  Request Processor
    7)  Load-Balancing Algorithm  15)  Subtree View
    8)  Location                  16)  Work Queue

    o)  'Basic' objects are shown - change this
    q)  quit

Enter choice: 15
```

2. On the Subtree View management menu, enter the number corresponding to "Create a new Subtree View".

```
>>>> Subtree View management menu

What would you like to do?

    1)  List existing Subtree Views
    2)  Create a new Subtree View
    3)  View and edit an existing Subtree View
    4)  Delete an existing Subtree View

    b)  back
    q)  quit

Enter choice [b]: 2
```

3. Enter `t` to base our new subtree view on the one created previously.

```
>>>> Choose how to create the new Subtree View:

    n)  new Subtree View created from scratch
    t)  use an existing Subtree View as a template

    b)  back
    q)  quit

Enter choice [n]: t
```

4. Select the `dc_example_dc_com-view` subtree view.

```
>>>> Select an existing Subtree View to use as a template for
the new Subtree View configuration:

    1)  dc_example_dc_com-view
    n)  new Proxy Transformation created from scratch
    c)  cancel
    q)  quit

Enter choice [n]: 1
```

5. Enter a descriptive name for the subtree view configuration.

```
>>>> Enter a name for the Subtree View that you want to create:
o_sample-view
```

6. Configure the base DN property of the Sample dataset.

```
>>>> Enter a value for the 'base-dn' property: o=sample
```

7. Enter the request processor we created in the previous section.

```
>>>> Enter a value for the 'request-processor' property:
o_sample-req-processor
```

8. Review the configuration properties, and enter `f` to save our changes.

```
>>>> Configure the properties of the Subtree View

        Property           Value(s)
        ---------------------------------------------
    1)  description        -
    2)  base-dn            "o=sample"
    3)  request-processor  o_sample-req-processor

    ?)  help
    f)  finish - create the new Subtree View
```

```
    d)  display the equivalent dsconfig arguments to create this
        object
    b)  back
    q) quit

Enter choice [b]: f

The current server has been configured to keep its configuration
synchronized with all servers in the 'all-servers' server group,
which is composed of servers with ids [proxy-east-01:389,
proxy-east-02:389]. When applying a change to a group of servers,
every server must be available and validate the change before it's
applied to any server. Do you wish to update all servers in the
group or only the currently selected server?

   g) Update all servers in the ’all-servers’ group
   s) Update only the current server
   b) back
   c) cancel

Enter choice: g

The Subtree View was created successfully
```

Editing the Client Connection Policy

Finally, we edit the client connection policy to add our new `o=sample` subtree view.

To Edit the Client Connection Policy

1. On the UnboundID® Directory Proxy Server configuration console main menu, enter the number corresponding to **Client Connection Policy**.

```
>>>> UnboundID® Directory Proxy Server configuration console main menu

What do you want to configure?

    1)  Client Connection Policy  9)   Log Publisher
    2)  Connection Handler        10)  Log Retention Policy
    3)  External Server           11)  Log Rotation Policy
    4)  Global Attribute Index    12)  Placement Algorithm
    5)  Global Configuration      13)  Proxy Transformation
    6)  LDAP Health Check         14)  Request Processor
    7)  Load-Balancing Algorithm  15)  Subtree View
    8)  Location                  16)  Work Queue

    o)  'Basic' objects are shown - change this
    q)  quit

Enter choice: 1
```

2. On the Client Connection management menu, enter the number corresponding to "Create a new Client Connection".

```
>>>> Client Connection Policy management 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
```

```
>>>> There is only one Client Connection Policy: 'default'. Are
you sure that this is the correct one? (yes / no) [yes]:
```

3. In the configuration properties,we select the `subtree-view` property. Enter the number corresponding to "Add one or more values" to add the new subtree view that we created for out example earlier.

```
>>>> Configuring the 'subtree-view' property

    Specifies the set of manually-configured subtree views
    that will be accessible to clients associated with this
    Client Connection Policy.

    If the base DN for a manually-configured subtree view
    conflicts with the base DN for an automatically-included
    backend subtree view (if the include-backend-subtree-views
    property has a value of true), then the manually-configured
    subtree view will be used rather than the automatically-included
    backend subtree view.

Do you want to modify the 'subtree-view' property?

    1)  Keep the value: dc_example_dc_com-view
    2)  Add one or more values
    3)  Remove one or more values
    4)  Leave undefined

    ?)  help
    q)  quit

Enter choice [1]: 2
```

4. Select the subtree view that was created in the previous section.

```
Select the Subtree Views you wish to add:

    1)  o_sample-view
    2)  Create a new Subtree View

    ?)  help
    b)  back
    q)  quit

Enter one or more choices separated by commas [b]:
```

5. Review the subtree views now referenced by the property and press **Enter** to use these values.

```
>>>> Configuring the 'subtree-view' property (Continued)

The 'subtree-view' property references the following Subtree
Views:

    *)  dc_example_dc_com-view
    *)  o_sample-view

Do you want to modify the 'subtree-view' property?

    1)  Use these values
    2)  Add one or more values
    3)  Remove one or more values
    4)  Leave undefined
    5)  Revert changes

    ?)  help
    q)  quit

Enter choice [1]:
```

6. Review the configuration properties of the client connection policy and enter `f` to save our changes.

```
>>>> Configure the properties of the Client Connection Policy

         Property                                  Value(s)
         ----------------------------------------------------------
    1)   policy-id                                 default
    2)   description                               -
    3)   enabled                                   true
    4)   evaluation-order-index                    9999
    5)   connection-criteria                       -
    6)   allowed-operation                         abandon, add, bind
                                                   compare, delete,
                                                   extended, modify,
                                                   modify-dn, search
    7)   allowed-extended-operation                -
    8)   denied-extended-operation                 -
    9)   allowed-auth-type                         sasl, simple
    10)  allowed-sasl-mechanism                    -
    11)  denied-sasl-mechanism                     -
    12)  include-backend-subtree-views             true
    13)  included-backend-base-dn                  -
    14)  excluded-backend-base-dn                  -
    15)  include-backend-server-passthrough-subtree-views  true
    16)  subtree-view                              dc_example_dc_
                                                   com-view,
                                                   o_sample-view

    ?)   help
    f)   finish - apply any changes to the Client Connection Policy
    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

The current server has been configured to keep its configuration
synchronized with all servers in the 'all-servers' server group,
which is composed of servers with ids [proxy-east-01:389,
proxy-east-02:389]. When applying a change to a group of servers,
every server must be available and validate the change before it's
applied to any server. Do you wish to update all servers in the
group or only the currently selected server?

   g) Update all servers in the ’all-servers’ group
   s) Update only the current server
   b) back
   c) cancel

Enter choice: g

The Client Connection Policy was modified successfully
```

Testing Proxy Transformations

After setting up the deployment scenario, the proxy server will now respond to requests to the `dc=example,dc=com` and o=sample base DNs. We now test the service by imitating example client requests to search and modify users.

Testing Proxy Transformations

The following example fetches the user with sampleID=scase under the `ou=east,o=sample` base DN.

1. Run `ldapsearch` to view a Sample entry.

```
root@proxy-east-01: bin/ldapsearch --bindDN "cn=directory manager" \
--bindPassword password --baseDN "ou=east,o=sample" "(sampleID=scase)"
```

```
dn: sampleID=scase,ou=People,ou=east,o=sample
objectClass: person
objectClass: organizationalPerson
objectClass: inetOrgPerson
objectClass: exampleAccount
objectClass: top
description: A customer account migrated from Sample merger
sampleID: scase
userPassword: {SSHA}A5O4RrQHWXc2Ii3btD4exGdP0TVW9VL3CR3ZXA==
exAccountNumber: 234098
givenName: Sterling
cn: Sterling Case
sn: Case
telephoneNumber: +1 804 094 3356
street: 00468 Second Street
mail: sterlingcase@maildomain.com
l: Arlington
st: VA
sampleRegion: east
sampleLinkedAccounts: sampleID=jcase,ou=People,ou=east,o=sample
```

2. Modify the sampleRegion value, changing it to west. To do this, we first create a `ldapmodify` input file, called `scase-mod.ldif`, with the following contents:

```
dn: sampleID=scase,ou=People,ou=east,o=sample
changetype: modify
replace: sampleRegion
sampleRegion: west
```

3. Use the file as an argument in the `ldapmodify` command as follows.

```
root@proxy-east-01: bin/ldapmodify --bindDN "cn=Directory Manager" \
--bindPassword password --filename scase-mod.ldif
```

```
Processing MODIFY request for sampleID=scase,ou=People, ou=east,o=sample
MODIFY operation successful for DN sampleID=scase,ou=People, ou=east,o=sample
```

4. Search for scase's `sampleRegion` value under o=sample, we should see west:

```
root@proxy-east-01: bin/ldapsearch --bindDN "cn=directory manager" \
--bindPassword password --baseDN "o=sample"  "(sampleID=scase)" \
sampleRegion
```

```
dn: sampleID=scase,ou=People,ou=east,o=sample
sampleRegion: west
```

5. Search for scase by `uid` rather than `sampleID`, under the `dc=example,dc=com` base DN. We see the Example.com schema version of the entry:

```
root@proxy-east-01: bin/ldapsearch --bindDN "cn=directory manager" \
--bindPassword password --baseDN "dc=example,dc=com"  "(uid=scase)"
```

```
dn: uid=scase,ou=People,dc=example,dc=com
objectClass: person
objectClass: exampleAccount
objectClass: inetOrgPerson
objectClass: organizationalPerson
objectClass: top
description: A customer account migrated from Sample merger
uid: scase
userPassword: {SSHA}A5O4RrQHWXc2Ii3btD4exGdP0TVW9VL3CR3ZXA==
exAccountNumber: 234098
givenName: Sterling
cn: Sterling Case
sn: Case
telephoneNumber: +1 804 094 3356
```

```
street: 00468 Second Street
mail: sterlingcase@maildomain.com
l: Arlington
st: VA
exSampleRegion: west
exSampleLinkedAccounts: uid=jcase,ou=People,dc=example,dc=com
```

Deploying an Entry-Balancing Proxy Configuration

Entry-balancing is a directory proxy server configuration that allows the entries within a portion of the directory information tree (DIT) to reside on multiple external servers. This configuration is typically useful when the DIT contains many millions of entries, which can be difficult to bring completely into memory for optimal performance. Entry-balancing allows entries under a balancing point base DN to be divided among any number of separate directory servers, making the directory proxy server responsible for intelligently routing requests based on the division.

In this example scenario, the entries in the DIT outside of the balancing point are replicated across all external servers known to the directory proxy server. Replication on the external directory servers must be properly configured before proceeding through this example. The directory servers are expected to contain two replication domains: the global domain, `dc=example,dc=com`, and the balancing point, `ou=people,dc=example,dc=com`.

In this deployment scenario, an austin-proxy1 instance of the directory proxy server communicates with four external directory servers. The directory proxy server is configured to use entry balancing for the `ou=people,dc=example,dc=com` base DN, with two sets of user entries split beneath it. The first set of user entries is defined in the replicated pair of external servers, `austin-set1.example.com` and `newyork-set1.example.com`. The second set of entries is defined in `austin-set2.example.com` and `newyork-set2.example.com`. The entries in the dc=example,dc=com DIT outside of the balancing point base DN are replicated among the four external servers.

The following `dsreplication status` output from the UnboundID® Directory Server external servers describes the replication configuration that exists before creating the directory proxy server configuration.

```
--- Replication Status for dc=example,dc=com: Enabled ---

Server : Entries : Backlog : Oldest Backlog Change Age : Generation ID
--------:---------:--------:---------------------------:-------------

austin-set1.example.com:389  : 10003   : 0   : N/A  : 722087263
austin-set2.example.com:389  : 10003   : 0   : N/A  : 722087263
newyork-set1.example.com:389 : 10003   : 0   : N/A  : 722087263
newyork-set2.example.com:389 : 10003   : 0   : N/A  : 722087263

--- Replication Status for ou=people,dc=example,dc=com (Set: dataset1): Enabled ---

Server : Entries : Backlog : Oldest Backlog Change Age : Generation ID
--------:---------:---------:---------------------------:-----------------

austin-set1.example.com:389  : 100001  : 0  : N/A               : 178892712
newyork-set1.example.com:389 : 100001  : 0  : N/A               : 178892712

--- Replication Status for ou=people,dc=example,dc=com (Set: dataset2): Enabled ---

Server : Entries : Backlog : Oldest Backlog Change Age : Generation ID
---------:---------:---------:---------------------------:-----------------

austin-set2.example.com:389  : 100001  : 0       : N/A        : 1057593890
newyork-set2.example.com:389 : 100001  : 0       : N/A        : 1057593890
```

Overview of Deployment Steps

In this deployment scenario, we will take the following steps:

- Install the proxy server on austin-proxy1.
- Use the `create-initial-proxy-config` tool to provide our initial setup for entry-balancing. The initial setup includes defining multiple subtree views and global indexes in support of entry balancing.
- Change the placement algorithm of the austin-proxy-01 server to use an entry-count placement algorithm. This algorithm is used to select the backend set to which to forward an add request. It looks at the number of entries in the backend sets and forwards the add request to the backend with either the fewest or the most entries, depending on the configuration. You can also configure the placement algorithm to make the decision based on the on-disk database size rather than the number of entries.

Installing Directory Proxy Server

We start by configuring the directory proxy server. The four external servers, `austin-set1.example.com`, `newyork-set1.example.com`, `austin-set2.example.com`, and `newyork-set2.example.com`, are running.

To Install the Directory Proxy Server

- Run the `setup` program in non-interactive mode.

```
root@austin-proxy1: ./setup --acceptLicense \
--listenAddress austin-proxy1.example.com \
--ldapPort 389 --rootUserDN "cn=Directory Manager" \
--rootUserPassword pass --entryBalancing \
--aggressiveJVMTuning --maxHeapSize 2g --no-prompt
```

Configuring the Entry Balancing Directory Proxy Server

Once the directory proxy server has been installed, it can be automatically configured using the `create-initial-proxy-config` tool. This tool can only be used once for this initial configuration, after which we will have to use `dsconfig` to make any changes to our directory proxy server configuration.

To Configure the Entry-Balancing Directory Proxy Server

1. Run the `create-initial-proxy-config` tool.

```
root@austin-proxy1: ./bin/create-initial-proxy-config
```

2. Our topology meets the requirements, press **Enter** to continue:

```
Some assumptions are made about the topology to keep
this tool simple:

1) all servers will be accessible via a single user account
2) all servers support the same communication security type
3) all servers are UnboundID-Proxy 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 web console to fine tune the configuration.

Would you like to continue? (yes / no) [yes]:
```

3. Provide the external server access credentials. All of our directory proxy servers have identical proxy user accounts and passwords.

```
Enter the DN of the proxy user account [cn=Proxy User,cn=Root DNs,cn=config]:

Enter the password for 'cn=Proxy User,cn=Root DNs,cn=config':
Confirm the password for 'cn=Proxy User,cn=Root DNs,cn=config':
```

4. Specify the type of security we are using. In this example, we do not use 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. First enter the dc=example,dc=com base DN as a non-entry-balancing base DN.

```
Enter a base DN of the Directory Server instances that will be
accessed through the Directory Proxy Server:

    b)  back
    q)  quit

Enter a DN or choose a menu item [dc=example,dc=com]:
```

6. Define the balancing point as a separate base DN, which is entry balanced:

```
Enter another base DN of the directory server instances that
will be accessed through the Directory Proxy Server:

    1)Remove dc=example,dc=com

    b)back

    q)quit

Enter a DN or choose a menu item [Press ENTER when finished
entering base DNs]: ou=people,dc=example,dc=com

Are entries within 'ou=people,dc=example,dc=com' split across
multiple servers so that each server stores only a subset of
the entries (i.e. is this base DN 'entry balanced')? (yes / no)
[no]: yes
```

7. In this example, the data in `ou=people,dc=example,dc=com` will be split across two backend sets. Enter 2 to specify that the data will be balanced across two sets of servers.

```
Across how many sets of servers is the data balanced?
```

```
    c) cancel creating ou=people,dc=example,dc=com
    q) quit

Enter a number greater than one or choose a menu item: 2
```

8. The balancing point is the same as our base DN, ou=people,dc=example,dc=com., so we use it as the entry balancing base.

```
>>>> Entry Balancing Base

The entry balancing base DN specifies the entry below which the
data is balanced. Entries not below this entry must be duplicated
in all the server sets. If all the entries in the base DN are
distributed the entry balancing base DN is the same as the base DN.

    c) cancel creating ou=people,dc=example,dc=com
    b) back
    q) quit

Enter the entry balancing base DN or choose a menu item
[ou=people,dc=example,dc=com]: ou=people,dc=example,dc=com
```

9. To improve the performance for equality search filters referencing the `uid` attribute, we create a `uid` global index. Enter `yes` to add a new attribute.

```
Would you like to add attributes to the global index? (yes / no) [no]: yes
```

10. Specify the `uid` attribute.

```
Enter attributes that you would like to add to the global index:

    c)cancel creating ou=people,dc=example,dc=com
    b)back
    q)quit

Enter an attribute name or choose a menu item [Press ENTER when
finished entering index attributes]: uid
```

11. To optimize directory proxy server performance from the moment it starts accepting connections, enter the number corresponding to "Yes, and all subsequent attributes":

```
Should the index for attribute 'uid' be primed such that it is
loaded into memory before the Directory Proxy Server begins
accepting connections?

    1) Yes
    2) No
    3) Yes, and all subsequent attributes
    4) No, and all subsequent attributes

Enter choice [1]: 3

Are the values of 'uid' guaranteed to be unique? (yes / no) [no]: yes
```

12. Press **Enter** to finish specifying index attributes.

```
Enter attributes that you would like to add to the global index:

    1) Remove uid (primed,unique)

    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]:
```

13. Press **Enter** to enable RDN index priming.

```
Would you like to enable RDN index priming for
'ou=people,dc=example,dc=com'? (yes / no) [yes]:
```

14. Press **Enter** to finish specifying base DNs.

```
Enter another base DN of the directory server instances that
will be accessed through the Directory Proxy Server:

    1) Remove dc=example,dc=com
    2) Remove ou=people,dc=example,dc=com (distributed)

    b)  back
    q)  quit

Enter a DN or choose a menu item [Press ENTER when finished
entering base DNs]:
```

15. The external servers are spread among two locations, New York and Austin. This proxy server instance is located in the austin location.

```
A good rule of thumb when naming locations is to use the
name of your data centers or the cities containing them.

    b)  back
    q)  quit

Enter a location name or choose a menu item: austin

    1)  Remove austin

    b)  back
    q)  quit
```

16. Define the newyork location:

```
Enter another location name or choose a menu item [Press ENTER
when finished entering locations]: newyork

    1)  Remove austin
    2)  Remove newyork

    b)  back
    q)  quit

Enter another location name or choose a menu item [Press ENTER
when finished entering locations]:
```

17. Select the austin location for this directory proxy server instance:

```
Choose the location for this Directory Proxy Server

    1) austin
    2) newyork

    b) back
    q) quit

Enter choice [1]:
```

18. Specify the LDAP external server instances associated with this location.

```
Enter the host and port (host:port) of the first directory server
in 'austin'

     b)  back
     q)  quit
```

```
Enter a host:port or choose a menu item [localhost:389]:
austin-set1.example.com:389
```

19. Specify that the `austin-set1` server can handle requests from the global domain and from set 1 restricted domain.

```
Assign server austin-set1.example.com:389 to handle requests for
one or more of the defined sets of data:

    1) dc=example,dc=com
    2) ou=people,dc=example,dc=com; Server Set 1
    3) ou=people,dc=example,dc=com; Server Set 2

Enter one or more choices separated by commas: 1,2
```

20. Enter the number corresponding to "Yes, and all subsequent servers" to prepare the server for access by the directory proxy server.

```
Would you like to prepare austin-set1.example.com:389 for access
by the Directory Proxy Server?

       1)Yes
       2)No
       3)Yes, and all subsequent servers
       4)No, and all subsequent servers

Enter choice [3]:
```

21. Select the entry-balanced data set that the austin-set1 server replicates with other servers.

```
You may choose a single entry-balanced data set with which
austin-set1.example.com:389 will replicate data with other servers

    1) ou=people,dc=example,dc=com; Server Set 1
    2) None, data will not be replicated

Enter choice: 1

Testing connection to austin-set1.example.com:389 ..... Done
Testing 'cn=Proxy User,cn=Root DNs,cn=config' access ....Denied
```

22. Modify the root user for use by the directory proxy server, specifying the directory manager password for the initial creation of the proxy user.

```
Would you like to create or modify root user 'cn=Proxy User,
cn=Root DNs,cn=config' so that it is available for this
Directory Proxy Server? (yes / no) [yes]:

Enter the DN of an account on austin-set1.example.com:389
with which to create or manage the 'cn=Proxy User,cn=Root DNs,
cn=config' account and configuration [cn=Directory Manager]:

Enter the password for 'cn=Directory Manager':
Created 'cn=Proxy User,cn=Root DNs,cn=config'
Testing 'cn=Proxy User,cn=Root DNs,cn=config'privileges...Done
Setting replication set name .....
```

23. Since the replication set name has already been configured, we do not need to use the name created automatically by the directory proxy server.

```
This server is currently configured for replication set 'dataset1'.
Would you like to reconfigure this server for replication set
'set-1'? (yes / no) [no]:

Setting replication set name ..... Done
Verifying backend 'dc=example,dc=com' ..... Done
Verifying backend 'ou=people,dc=example,dc=com' ..... Done
Testing 'cn=Proxy User' privileges ..... Done
```

```
Verifying backend 'dc=example,dc=com' ..... Done
```

24. Define the other Austin and New York servers using the same procedure as in the previous example:

```
Enter another server in 'austin'

    1) Remove austin-set1.example.com:389
    b) back
    q) quit

Enter a host:port or choose a menu item [Press ENTER when
finished entering servers]: austin-set2.example.com:389

Assign server austin-set2.example.com:389 to handle requests
for one or more of the defined sets of data

    1) dc=example,dc=com
    2) ou=people,dc=example,dc=com; Server Set 1
    3) ou=people,dc=example,dc=com; Server Set 2

Enter one or more choices separated by commas: 1,3

You may choose a single entry-balanced data set with which
austin-set2.example.com:389 will replicate data with other
servers

    1) ou=people,dc=example,dc=com; Server Set 2
    2) None, data will not be replicated

Enter choice: 1

Testing connection to austin-set2.example.com:389 ....Done
Testing 'cn=Proxy User,cn=Root DNs,cn=config' access ... Denied

Would you like to create or modify root user 'cn=Proxy User,
cn=Root DNs,cn=config' so that it is available for this
Directory Proxy Server? (yes / no) [yes]:

Would you like to use the previously entered manager credentials
to access all prepared servers? (yes / no) [yes]:

Created 'cn=Proxy User,cn=Root DNs,cn=config'
Testing 'cn=Proxy User,cn=Root DNs,cn=config' privileges...Done
Setting replication set name .....

This server is currently configured for replication set 'dataset2'.

Would you like to reconfigure this server for replication set 'set-2'?
(yes / no) [no]:

Setting replication set name ..... Done
Verifying backend 'dc=example,dc=com' ..... Done
Verifying backend 'ou=people,dc=example,dc=com' ..... Done

Enter another server in 'austin'

    1) Remove austin-set1.example.com:389
    2) Remove austin-set2.example.com:389

    b) back
    q) quit

Enter a host:port or choose a menu item [Press ENTER when
finished entering servers]:

>>>> >>>> Location 'newyork' Details
 >>>> External Servers

External Servers identify directory server instances including
host, port, and authentication information.

Enter the host and port (host:port) of the first directory server
in 'newyork':

    b) back
    q) quit
```

```
Enter a host:port or choose a menu item [localhost:389]:
newyork-set1.example.com:389

Assign server newyork-set1.example.com:389 to handle requests
for one or more of the defined sets of data

    1) dc=example,dc=com
    2) ou=people,dc=example,dc=com; Server Set 1
    3) ou=people,dc=example,dc=com; Server Set 2

Enter one or more choices separated by commas: 1,2

You may choose a single entry-balanced data set with which
newyork-set1.example.com:389 will replicate data with other servers

    1) ou=people,dc=example,dc=com; Server Set 1
    2) None, data will not be replicated

Enter choice: 1

Testing connection to newyork-set1.example.com:389 ....Done
Testing 'cn=Proxy User,cn=Root DNs,cn=config' access ... Denied

Would you like to create or modify root user 'cn=Proxy User,
cn=Root DNs,cn=config' so that it is available for this
Directory Proxy Server? (yes / no) [yes]:

Created 'cn=Proxy User,cn=Root DNs,cn=config'
Testing 'cn=Proxy User,cn=Root DNs,cn=config' privileges...Done
Setting replication set name .....

This server is currently configured for replication set 'dataset1'.

Would you like to reconfigure this server for replication set
'set-1'? (yes / no) [no]:

Setting replication set name ..... Done
Verifying backend 'dc=example,dc=com' ..... Done
Verifying backend 'ou=people,dc=example,dc=com' ..... Done

Enter another server in 'newyork'

    1) Remove newyork-set1.example.com:389
    b) back
    q) quit

Enter a host:port or choose a menu item [Press ENTER when
finished entering servers]: newyork-set2.example.com:389

Assign server newyork-set2.example.com:389 to handle requests
for one or more of the defined sets of data:

    1) dc=example,dc=com
    2) ou=people,dc=example,dc=com; Server Set 1
    3) ou=people,dc=example,dc=com; Server Set 2

Enter one or more choices separated by commas: 1,3

You may choose a single entry-balanced data set with which
new-york-set2.example.com:389 will replicate data with other servers

    1) ou=people,dc=example,dc=com; Server Set 2
    2) None, data will not be replicated

Enter choice: 1

Testing connection to newyork-set2.example.com:389 ..... Done
Testing 'cn=Proxy User,cn=Root DNs,cn=config' access.... Denied

Would you like to create or modify root user 'cn=Proxy User,
cn=Root DNs,cn=config' so that it is available for this Directory
Proxy Server? (yes / no) [yes]:

Created 'cn=Proxy User,cn=Root DNs,cn=config' Testing
'cn=Proxy User,cn=Root DNs,cn=config' privileges...Done
Setting replication set name .....

This server is currently configured for replication set 'dataset2'.
Would you like to reconfigure this server for replication
```

```
set 'set-2'? (yes / no) [no]:

Setting replication set name ..... Done
Verifying backend 'dc=example,dc=com' ..... Done
Verifying backend 'ou=people,dc=example,dc=com' ..... Done

Enter another server in 'newyork'

    1)Remove newyork-set1.example.com:389
    2)Remove newyork-set2.example.com:389

    b)back
    q)quit

Enter a host:port or choose a menu item [Press ENTER when
finished entering servers]:

>>>> >>>> Configuration Summary

  External Server Security: None
  Proxy User DN: cn=Proxy User,cn=Root DNs,cn=config
  Location austin
    Failover Order: newyork
    Servers: austin-set1.example.com:389,
             austin-set2.example.com:389
  Location newyork
    Failover Order: austin
    Servers: newyork-set1.example.com:389,
             newyork-set2.example.com:389
  Base DN: dc=example,dc=com
    Servers: austin-set1.example.com:389,
             austin-set2.example.com:389,
             newyork-set1.example.com:389,
             newyork-set2.example.com:389
  Base DN:vou=people,dc=example,dc=com
    Entry Balancing Base: ou=people,dc=example,dc=com
    Server Set 1: austin-set1.example.com:389,
                  newyork-set1.example.com:389
    Server Set 2: austin-set2.example.com:389,
                  newyork-set2.example.com:389
    Index Attributes: uid (primed,unique)
    Prime RDN Index: Yes

    NOTE: The Directory Proxy Server must be restarted after
    this tool has completed to have index priming take place

       b) back
       q) quit
       w) write configuration

    Enter choice [w]
    >>>> Write Configuration

    The configuration will be written to a 'dsconfig' batch
    file that can be used to configure other Directory Proxy Servers

    Writing Directory Proxy Server configuration to /proxy/dps-cfg.txt.....Done
```

25. Enter `yes` to apply our configuration changes to the directory proxy server.

```
Apply these configuration changes to the local Directory Proxy
Server? (yes /no) [yes]:

How do you want to connect to the Directory Proxy Server on localhost?

    1) LDAP
    2) LDAP with SSL
    3) LDAP with StartTLS

Enter choice [1]:

Administrator user bind DN [cn=Directory Manager]:
Password for user 'cn=Directory Manager':
Creating Locations ..... Done
Updating Failover Locations ..... Done
Updating Global Configuration ..... Done
Creating Health Checks ..... Done
Creating External Servers ..... Done
```

```
Creating Load-Balancing Algorithm for dc=example,dc=com .... Done
Creating Request Processor for dc=example,dc=com ..... Done
Creating Subtree View for dc=example,dc=com ..... Done
Updating Client Connection Policy for dc=example,dc=com ..... Done
Creating Load-Balancing Algorithm for ou=people,dc=example,dc=com; Server Set 1 .....
 Done
Creating Request Processor for ou=people,dc=example,dc=com; Server Set 1...Done
Creating Load-Balancing Algorithm for ou=people,dc=example,dc=com; Server Set 2 ....
 Done
Creating Request Processor for ou=people,dc=example,dc=com; Server Set 2...Done
Creating Entry Balancing Request Processor for ou=people,dc=example,dc=com ..... Done
Creating Placement Algorithm for ou=people,dc=example,dc=com .... Done
Creating Global Attribute Indexes for ou=people,dc=example,dc=com ..... Done
Creating Subtree View for ou=people,dc=example,dc=com ..... Done
Updating Client Connection Policy for ou=people,dc=example,dc=com ..... Done

See /logs/create-initial-proxy-config.log for a detailed log of this operation

To see basic server configuration status and configuration you can launch /bin/status
```

Configuring the Placement Algorithm Using a Batch File

Now, we configure the placement algorithm using a batch file. We want to place new entries added through the proxy via LDAP ADD operations into the least used dataset. We do this using an entry-count placement algorithm. To change the placement algorithm from round robin to entry count, we first create and enable an entry-count placement algorithm configuration object and then disable the existing round robin placement algorithm. Our batch file, `dsconfig.post-setup`, contains the `dsconfig` commands required to create the entry-count placement algorithm and disable the old round robin algorithm.

To Configure the Placement Algorithm Using a Batch File

The batch file contains comments to explain each `dsconfig` command. Note that in this example, line wrapping is used for clarity. The dsconfig command requires that the full command be provided on a single line.

The batch file itself looks like the following:

```
root@austin-proxy1:more ../dsconfig.post-setup

# This dsconfig operation creates the entry-count placement
# algorithm with the default behavior of adding entries to the
# smallest backend dataset first.

dsconfig create-placement-algorithm
--processor-name ou_people_dc_example_dc_com-eb-req-processor
--algorithm-name entry-count --type entry-counter --set enabled:true

# Note that once the entry-count placement algorithm is created
# and enabled, we can disable the round-robin algorithm.
# Since an entry-balancing proxy must always have a placement
# algorithm, we add a second algorithm and then disable the
# original round-robin algorithm created during the setup
# procedure.

dsconfig set-placement-algorithm-prop
--processor-name ou_people_dc_example_dc_com-eb-req-processor
--algorithm-name round-robin --set enabled:false

# At this point, LDAP ADD operations will be forwarded to an external
# server representing the dataset with the least number of entries.
```

- Run the `dsconfig` command using the batch file. Once the batch file has executed, a new entry-count placement algorithm, called entry-count, has been created, and the old round-robin placement algorithm, `round-robin`, has been disabled.

```
root@austin-proxy1: bin/dsconfig --no-prompt \
--bindDN "cn=directory manager" --bindPassword password \
--port 389  --batch-file ../dsconfig.post-setup
```

```
Batch file '../dsconfig.post-setup' contains 2 commands

Executing: create-placement-algorithm --no-prompt
--bindDN "cn=directory manager" --bindPassword pass
--port 1389
--processor-name ou_people_dc_example_dc_com-eb-req-processor
--algorithm-name entry-count --type entry-counter --set enabled:true

Executing: delete-placement-algorithm --no-prompt
--bindDN "cn=directory manager" --bindPassword pass
--port 1389
--processor-name ou_people_dc_example_dc_com-eb-req-processor
--algorithm-name round-robin --set enabled:false
```

Chapter 8 Troubleshooting the Directory Proxy Server

This chapter provides the common problems and potential solutions that might occur when running UnboundID® Directory Proxy Server. It is primarily targeted at cases in which the directory proxy server is running on SolarisTM or Linux® systems, but much of the information can be useful on other platforms as well.

This chapter presents the following information:

Topics:

Garbage Collection Diagnostic Information

You can enable the JVM debugging options to track garbage collection data for your system. The options can impact JVM performance, but they provide valuable data to tune your server when troubleshooting garbage collection issues. While the `jstat` utility with the `-gc` option can be used to obtain some information about garbage collection activity, there are additional arguments that can be added to the JVM to use when running the server to provide additional detail.

```
-XX:+PrintGCDetails
-XX:+PrintTenuringDistribution
-XX:+PrintGCApplicationConcurrentTime
-XX:+PrintGCApplicationStoppedTime
-XX:+PrintGCDateStamps
```

To run the Directory Proxy Server with these options, edit the `config/java.properties` file and add them to the end of the line that begins with "`bin/start-proxy.java-args`". After the file has been saved, invoke the following command to make those new arguments take effect the next time the server is started:

```
$ bin/dsjavaproperties
```

Working with the Troubleshooting Tools

If problems arise with the directory proxy server (whether from issues in the directory proxy server itself or a supporting component, like the JVM, operating system, or hardware), then it is essential to be able to diagnose the problem quickly to determine the underlying cause and the best course of action to take towards resolving it.

Working with the Collect Support Data Tool

The Directory Proxy Server provides a significant amount of information about its current state including any problems that it has encountered during processing. If a problem occurs, the first step is to run the `collect-support-data` tool in the `bin` directory. The tool aggregates all relevant support files into a zip file that administrators can send to your authorized support provider for analysis. The tool also runs data collector utilities, such as `jps`, `jstack`, and `jstat` plus other diagnostic tools for Solaris and Linux machines, and bundles the results in the zip file.

The tool may only archive portions of certain log files to conserve space, so that the resulting support archive does not exceed the typical size limits associated with e-mail attachments.

The data collected by the `collect-support-data` tool varies between systems. For example, on Solaris Zone, configuration information is gathered using commands like `zonename` and `zoneadm`. However, the tool always tries to get the same information across all systems for the target Directory Proxy Server. The data collected includes the configuration directory, summaries and snippets from the `logs` directory, an LDIF of the monitor and RootDSE entries, and a list of all files in the server root.

Available Tool Options

The `collect-support-data` tool has some important options that you should be aware of:

- **--noLdap**. Specifies that no effort should be made to collect any information over LDAP. This option should only be used if the server is completely unresponsive or will not start and only as a last resort.
- **--pid {pid}**. Specifies the ID of an additional process from which information is to be collected. This option is useful for troubleshooting external server tools and can be specified multiple times for each external server, respectively.
- **--sequential**. Use this option to diagnose “Out of Memory” errors. The tool collects data in parallel to minimize the collection time necessary for some analysis utilities. This option specifies that data collection should be run sequentially as opposed to in parallel. This action has the effect of reducing the initial memory footprint of this tool at a cost of taking longer to complete.
- **--reportCount {count}**. Specifies the number of reports generated for commands that supports sampling (for example, `vmstat`, `iostat`, or `mpstat`). A value of 0 (zero) indicates that no reports will be generated for these commands. If this option is not specified, it defaults to 10.
- **--reportInterval {interval}**. Specifies the number of seconds between reports for commands that support sampling (for example, `mpstat`). This option must have a value greater than 0 (zero). If this option is not specified, it default to 1.
- **--maxJstacks {number}**. Specifies the number of jstack samples to collect. If not specified, the default number of samples collected is 10.
- **--collectExpensiveData**. Specifies that data on expensive or long running processes be collected. These processes are not collected by default, because they may impact the performance of a running server.
- **--comment {comment}**. Provides the ability to submit any additional information about the collected data set. The comment will be added to the generated archive as a README file.
- **--includeBinaryFiles**. Specifies that binary files be included in the archive collection. By default, all binary files are automatically excluded in data collection.
- **--adminPassword {adminPassword}**. Specifies the global administrator password used to obtain `dsreplication status` information.
- **--adminPasswordFile {adminPasswordFile}**. Specifies the file containing the password of the global administrator used to obtain `dsreplication status` information.

To Run the Collect Support Data Tool

1. Go to the server root directory.
2. Use the `collect-support-data` tool. Make sure to include the host, port number, bind DN, and bind password.

```
$ bin/collect-support-data --hostname 127.0.0.1 --port 389 \
  --bindDN "cn=Directory Manager" --bindPassword secret \
```

```
  --serverRoot /opt/UnboundID-Proxy --pid 1234
```

3. Email the zip file to your Authorized Support Provider.

Directory Proxy Server Troubleshooting Tools

The UnboundID® Directory Proxy Server provides a set of tools that can also be used to obtain information for diagnosing and solving problems.

Server Version Information

If it becomes necessary to contact your authorized support provider, then it will be important to provide precise information about the version of the Directory Proxy Server software that is in use. If the server is running, then this information can be obtained from the "`cn=Version,cn=monitor`" entry. It can also be obtained using the command:

```
$ bin/status --fullVersion
```

This command outputs a number of important pieces of information, including:

- Major, minor, point and patch version numbers for the server.
- Source revision number from which the server was built.
- Build information including build ID with time stamp, OS, user, Java and JVM version for the build.
- Auxiliary software versions: Jetty, JZlib, SNMP4j (SNMP4J, Agent, Agentx), Groovy, UnboundID LDAP SDK for Java, and UnboundID Server SDK.

LDIF Connection Handler

The Directory Proxy Server provides an LDIF connection handler that provides a way to request operations that do not require any network communication with the server. This can be particularly helpful if a configuration problem or bug in the server has left a connection handler unusable, or if all worker threads are busy processing operations.

The LDIF connection handler is enabled by default and looks for LDIF files to be placed in the `config/auto-process-ldif` directory. This directory does not exist by default, but if it is created and an LDIF file is placed in it that contains one or more changes to be processed, then those changes will be applied.

Any changes that can be made over LDAP can be applied through the LDIF connection handler. It is primarily intended for administrative operations like updating the server configuration or scheduling tasks, although other types of changes (including changes to data contained in the server) can be processed. As the LDIF file is processed, a new file is written with comments for each change providing information about the result of processing that change.

Embedded Profiler

If the Directory Proxy Server appears to be running slowly, then it is helpful to know what operations are being processed in the server. The JVM Stack Trace monitor entry can be used to obtain a point-in-time snapshot of what the server is doing, but in many cases, it might be useful to have information collected over a period of time.

The embedded profiler is configured so that it is always available but is not active by default so that it has no impact on the performance of the running server. Even when it is running, it has a relatively small impact on performance, but it is recommended that it remain inactive when it is not needed. It can be controlled using the `dsconfig` tool or the web administration console by managing the "Profiler" configuration object in the "Plugin" object type, available at the standard object level. The `profile-action` property for this configuration object can have one of the following values:

- **start** – Indicates that the embedded profiler should start capturing data in the background.
- **stop** – Indicates that the embedded profiler should stop capturing data and write the information that it has collected to a `logs/profile{timestamp}` file.
- **cancel** – Indicates that the embedded profiler should stop capturing data and discard any information that it has collected.

Any profiling data that has been captured can be examined using the `profiler-viewer` tool. This tool can operate in either a text-based mode, in which case it dumps a formatted text representation of the profile data to standard output, or it can be used in a graphical mode that allows the information to be more easily understood.

To Invoke the Profile Viewer in Text-based Mode

- Run the `profile-viewer` command and specify the captured log file using the `--fileName` option.

```
$ bin/profile-viewer --fileName logs/profile.20110101000000Z
```

To Invoke the Profile Viewer in GUI Mode

- Run the `profile-viewer` command and specify the captured log file using the `--fileName` option. To invoke GUI mode, add the option `--useGUI`.

```
$ bin/profile-viewer --fileName logs/profile.20110101000000Z --useGUI
```

Troubleshooting Resources for Java Applications

Because the UnboundID® Directory Proxy Server is written entirely in Java, it is possible to use standard Java debugging and instrumentation tools when troubleshooting problems with the Directory Proxy Server. In many cases, obtaining the full benefit of these tools requires

access to the Directory Proxy Server source code. These Java tools should be used under the advisement of your authorized support provider.

Java Troubleshooting Documentation (Oracle/Sun JDK)

There are a number of documents providing general information about troubleshooting Java-based applications. Some of these documents include:

- http://www.oracle.com/technetwork/java/javase/index-138283.html – Troubleshooting Java SE
- http://www.oracle.com/technetwork/java/javase/index-137495.html – Troubleshooting Guide for Java SE 6 with HotSpot VM
- http://www.sun.com/bigadmin/hubs/java/troubleshoot/ – BigAdmin Page on Java SE Troubleshooting
- http://www.oracle.com/technetwork/java/javase/tools6-unix-139447.html – Tools for troubleshooting Java on Solaris and Linux

Java Troubleshooting Tools (Oracle/Sun JDK)

The Java Development Kit provides a number of very useful tools to obtain information about Java applications and diagnosing problems. These tools are not included with the Java Runtime Environment (JRE), so the full Java Development Environment (JDK) should always be installed and used to run the UnboundID® Directory Proxy Server.

jps

The `jps` tool is a Java-specific version of the UNIX `ps` tool. It can be used to obtain a list of all Java processes currently running and their respective process identifiers. When invoked by a non-root user, it will list only Java processes running as that user. When invoked by a root user, then it lists all Java processes on the system.

This tool can be used to see if the Directory Proxy Server is running and if a process ID has been assigned to it. This process ID can be used in conjunction with other tools to perform further analysis.

This tool can be run without any arguments, but some of the more useful arguments that include:

- **-v** – Includes the arguments passed to the JVM for the processes that are listed.
- **-m** – Includes the arguments passed to the main method for the processes that are listed.
- **-l** – (lowercase L). Include the fully qualified name for the main class rather than only the base class name.

Additional documentation for the `jps` tool is available at:

- http://java.sun.com/javase/6/docs/techs/tools/share/jps.html

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 UnboundID® Directory Proxy Server), and the frame of the stack trace may include the line number of the source file to which it corresponds. This is useful when diagnosing problems and often allows them to be identified and resolved quickly.

To obtain a stack trace from a running JVM, use the command:

```
jstack {processID}
```

where {processID} is the process ID of the target JVM as returned by the `jps` command. To obtain a stack trace from a core file from a Java process, use the command:

```
jstack {pathToJava} {pathToCore}
```

where `{pathToJava}` is the path to the java command from which the core file was created, and `{pathToCore}` is the path to the core file to examine. In either case, the stack trace is written to standard output and includes the names and call stacks for each of the threads that were active in the JVM.

In many cases, no additional options are necessary. The "`-l`" option can be added to obtain a long listing, which includes additional information about locks owned by the threads. The "`-m`" option can be used to include native frames in the stack trace.

Additional documentation for the `jstack` tool is available at http://java.sun.com/javase/6/ docs/ techs/tools/share/jstack.html.

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.

Additional information about the `jmap` tool can be found at http://java.sun.com/javase/6/ docs/techs/tools/share/jmap.html.

jhat

The `jhat` (Java Heap Analysis Tool) utility provides the ability to analyze the contents of the Java heap. It can be used to analyze a heap dump file, which is generated if the Directory Proxy Server encounters an out of memory error (as a result of the "`-XX:+HeapDumpOnOutOfMemoryError`" JVM option) or from the use of the `jmap` command with the "`-dump`" option.

The `jhat` tool acts as a web server that can be accessed by a browser in order to query the contents of the heap. Several predefined queries are available to help determine the types of objects consuming significant amounts of heap space, and it also provides a custom query language (OQL, the Object Query Language) for performing more advanced types of analysis.

The `jhat` tool can be launched with the path to the heap dump file, like:

```
jhat /path/to/heap.dump
```

This command causes the `jhat` web server to begin listening on port 7000. It can be accessed in a browser at `http://localhost:7000` (or `http://address:7000` from a remote system). An alternate port number can be specified using the "`-port`" option, like:

```
jhat -port 1234 /path/to/heap.dump
```

To issue custom OQL searches, access the web interface using the URL `http://localhost:7000/oql/` (the trailing slash must be provided). Additional information about the OQL syntax may be obtained in the web interface at `http://localhost:7000/oqlhelp/`. Additional information for the `jhat` tool may be found at `http://java.sun.com/javase/6/docs/techs/tools/share/jhat.html`.

jstat

The `jstat` tool is used to obtain a variety of statistical information from the JVM, much like the `vmstat` utility that can be used to obtain CPU utilization information from the operating system. The general manner to invoke it is as follows:

```
jstat {type} {processID} {interval}
```

The `{interval}` option specifies the length of time in milliseconds between lines of output. The `{processID}` option specifies the process ID of the JVM used to run the Directory Proxy Server, which can be obtained by running `jps` as mentioned previously. The `{type}` option specifies the type of output that should be provided. Some of the most useful types include:

- **-class** – Provides information about class loading and unloading.
- **-compile** – Provides information about the activity of the JIT complex.

- **-printcompilation** – Provides information about JIT method compilation.
- **-gc** – Provides information about the activity of the garbage collector.
- **-gccapacity** – Provides information about memory region capacities.

Java Diagnostic Information

In addition to the tools listed in the previous section, the JVM can provide additional diagnostic information in response to certain events. UnboundID® Directory Proxy Server supports the Sun/Oracle JDK 1.6.0_31 and the IBM JRE 1.6.0 IBM J9 2.4 for 32-bit and 64-bit architectures.

Garbage Collection Diagnostic Information

You can enable the JVM debugging options to track garbage collection data for your system. The options can impact JVM performance, but they provide valuable data to tune your server when troubleshooting garbage collection issues. While the `jstat` utility with the `-gc` option can be used to obtain some information about garbage collection activity, there are additional arguments that can be added to the JVM to use when running the server to provide additional detail.

```
-XX:+PrintGCDetails
-XX:+PrintTenuringDistribution
-XX:+PrintGCApplicationConcurrentTime
-XX:+PrintGCApplicationStoppedTime
-XX:+PrintGCDateStamps
```

To run the Directory Proxy Server with these options, edit the `config/java.properties` file and add them to the end of the line that begins with "`bin/start-proxy.java-args`". After the file has been saved, invoke the following command to make those new arguments take effect the next time the server is started:

```
$ bin/dsjavaproperties
```

JVM Crash Diagnostic Information

If the JVM itself should happen to crash for some reason, then it generates a fatal error log with information about the state of the JVM at the time of the crash. By default, this file is named `hs_err_pid{processID}.log` and is written into the base directory of the Directory Proxy Server installation. This file includes information on the underlying cause of the JVM crash, information about the threads running and Java heap at the time of the crash, the options provided to the JVM, environment variables that were set, and information about the underlying system. More information about the content that may be written to this log file may be found at `http://java.sun.com/javase/6/webs/trouble/TSG-VM/html/felog.html`.

Java Troubleshooting Tools (IBM JDK)

The UnboundID® Directory Proxy Server can be run on machines using the IBM JDK. IBM provides Java monitoring and diagnostic tools that can assess JVM performance and

troubleshoot any Java application failures. The following tools are available for the IBM JDK. For more detailed information, see the IBM Developers web-site for a description of each tool:

- **Health Center Version 1.3**. Monitors Java applications running on the JDK. The tool provides profiling information for performance, memory usage, system environment, object allocations and other areas.
- **Memory Analyzer Version 1.1**. Analyzes Java heap memory using a system or heap dump snapshot of a Java process.
- **Garbage Collection and Memory Visualizer Version 2.6**. Fine-tunes Java performance by optimizing garbage collection performance, provides Java heap recommendations based on peak and average memory usage, and detects memory leaks and heap exhaustion.
- **Dump Analyzer Version 2.2**. Helps troubleshoot the cause of any application failure using an operating system dump. The tool detects any potential problems based on state, thread, stack information and error messages that were generated when the application failed.
- **Diagnostics Collector Version 1.0**. Collects diagnostic and context information during Java runtime processes that failed. The tool verifies your Java diagnostic configuration to ensure that disabled diagnostic analyzers are enabled to troubleshoot a problem.
- **IBM Diagnostic Tool Framework for Java Version 1.5**. Runs on dump data extracted by the `jextract` tool. The tool checks memory locations, Java threads, Java objects and other important diagnostic areas when the system dump was produced.

Troubleshooting Resources in the Operating System

The underlying operating system also provides a significant amount of information that can help diagnose issues that impact the performance and the stability of the Directory Proxy Server. In some cases, problems with the underlying system can be directly responsible for the issues seen with the Directory Proxy Server, and in others system, tools can help narrow down the cause of the problem.

Identifying Problems with the Underlying System

If the underlying system itself is experiencing problems, it can adversely impact the function of applications running on it. Places to look for problems in the underlying system include:

- The system log file (`/var/adm/messages` on Solaris and `/var/log/messages` on Linux). Information about faulted or degraded devices or other unusual system conditions are written there.
- On Solaris systems, if the fault management system has detected a problem with a system component, information about that problem is obtain by running the `fmdump` command.
- If the ZFS filesystem is in use, then the `zpool` status command provides information about read errors, write errors, or data checksum errors.

Monitoring System Data Using the Metrics Engine

The UnboundID Metrics Engine provides collection and storage of performance data from your server topology. You can use the System Utilization Monitor with the Metrics Engine to collect information about the host system CPU, disk, and network utilization on any platform except Linux. If you are not using the Metrics Engine, you do not need to use the system utilization monitor. When data is being collected, it periodically forks the process and executes commands.

For more information about using the System Utilization Monitor, refer to the data collection chapter of the UnboundID Metrics Engine documentation.

Examining CPU Utilization

Observing CPU utilization for the Directory Proxy Server process and the system as a whole provides clues as to the nature of the problem.

System-Wide CPU Utilization

To investigate CPU consumption of the system as a whole, use the `vmstat` command with a time interval in seconds, like:

```
vmstat 5
```

The specific output of this command varies between different operating systems, but it includes the percentage of the time the CPU was spent executing user-space code (user time), the percentage of time spent executing kernel-space code (system time), and the percentage of time not executing any code (idle time).

If the CPUs are spending most of their time executing user-space code, the available processors are being well-utilized. If performance is poor or the server is unresponsive, it can indicate that the Directory Proxy Server is not optimally tuned. If there is a high system time, it can indicate that the system is performing excessive disk and/or network I/O, or in some cases, there can be some other system-wide problem like an interrupt storm. If the system is mostly idle but the Directory Proxy Server is performing poorly or is unresponsive, there can be a resource constraint elsewhere (for example, waiting on disk or memory access, or excessive lock contention), or the JVM can be performing other tasks like stop-the-world garbage collection that cannot be run heavily in parallel.

Per-CPU Utilization

To investigate CPU consumption on a per-CPU basis, use the `mpstat` command with a time interval in seconds, like:

```
mpstat 5
```

On Linux systems, it might be necessary to add "`-P ALL`" to the command, like:

```
mpstat -P ALL 5
```

Among other things, this shows the percentage of time each CPU has spent in user time, system time, and idle time. If the overall CPU utilization is relatively low but `mpstat` reports that one CPU has a much higher utilization than the others, there might be a significant bottleneck within the server or the JVM might be performing certain types of garbage collection which cannot be run in parallel. On the other hand, if CPU utilization is relatively even across all CPUs, there is likely no such bottleneck and the issue might be elsewhere.

Per-Process Utilization

To investigate CPU consumption on a per-process basis, use the `prstat` tool on Solaris or the `top` utility on Linux. If a process other than the Java process used to run the Directory Proxy Server is consuming a significant amount of available CPU, it might be interfering with the ability of the Directory Proxy Server to run effectively.

If the `mpstat` command showed that one CPU was much more heavily utilized than the others, it might be useful to identify the thread with the highest CPU utilization as it is likely the one that is a bottleneck preventing other threads from processing. On Solaris, this can be achieved by using the `prstat` command with the "`-L`" option, like:

```
prstat -L -p {processID}
```

This command will cause each thread to be displayed on a separate line, with the LWPID (lightweight process identifier) displayed as the last item on each line, separated from the process name by a slash. The thread that is currently consuming the largest amount of CPU will be displayed at the top of the list, and the `pstack` command can be used to identify which thread is responsible.

Examining Disk Utilization

If the underlying system has a very high disk utilization, it can adversely impact Directory Proxy Server performance. It could delay the ability to read or write database files or write log files. It could also raise concerns for server stability if excessive disk I/O inhibits the ability of the cleaner threads to keep the database size under control.

The `iostat` tool may be used to obtain information about the disk activity on the system. On Solaris systems, this should be invoked using the "`-x`" and "`-n`" arguments, like:

```
iostat -x -n 5
```

On Linux systems, `iostat` should be invoked with the "`-x`" argument, like:

```
iostat -x 5
```

A number of different types of information will be displayed, but to obtain an initial feel for how busy the underlying disks are, look at the "%b" column on Solaris and the "%util" column on Linux. Both of these fields show the percentage of the time that the underlying disks are actively servicing I/O requests. A system with a high disk utilization likely exhibits poor Directory Proxy Server performance.

If the high disk utilization is on one or more disks that are used to provide swap space for the system, the system might not have enough free memory to process requests. As a result, it might have started swapping blocks of memory that have not been used recently to disk. This

can cause very poor server performance. It is important to ensure that the server is configured appropriately to avoid this condition. If this problem occurs on a regular basis, then the server is likely configured to use too much memory. If swapping is not normally a problem but it does arise, then check to see if there are any other processes running, which are consuming a significant amount of memory, and check for other potential causes of significant memory consumption (for example, large files in a `tmpfs` filesystem).

On Solaris systems using ZFS, you can use the `zpool iostat {interval}` command to obtain information about I/O activity on a per-pool basis. While this command provides a useful display of the number of read and write operations and the amount of data being read from and written to the disks, it does not actually show how busy the underlying disks. As a result, the `zpool iostat` command is generally not as useful as the traditional `iostat` command for identifying potential I/O bottlenecks.

Examining Process Details

There are a number of tools provided by the operating system that can help examine a process in detail.

ps

The standard `ps` tool can be used to provide a range of information about a particular process. For example, the command can be used to display the state of the process, the name of the user running the process, its process ID and parent process ID, the priority and nice value, resident and virtual memory sizes, the start time, the execution time, and the process name with arguments:

```
ps -fly -p {processID}
```

Note that for a process with a large number of arguments, the standard `ps` command displays only a limited set of the arguments based on available space in the terminal window. In that case, the BSD version of the `ps` command (available on Solaris as `/usr/ucb/ps`) can be used to obtain the full command with all arguments, like:

```
/usr/ucb/ps auxwww {processID}
```

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 and is available on Solaris, but the Sun Studio debugger `dbx` is generally preferred over `gdb` on Solaris.

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) on Solaris, you can use the `pfiles` command, like:

```
pfiles {processID}
```

On Linux systems, the `lsof` tool can be used, like:

```
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 like you can get with `pstack` or a debugger. For example, if a process is performing a significant amount of disk reads and/or writes, it can be useful to see which files are being accessed. Similarly, if a process is consistently exiting abnormally, then beginning tracing for that process just before it exits can help provide additional information that cannot be captured in a core file (and if the process is exiting rather than being terminated for an illegal operation, then no core file may be available).

On Solaris systems, the `dtrace` tool provides an unmatched mechanism for tracing the execution of a process in extremely powerful and flexible ways, but it is also relatively complex and describing its use is beyond the scope of this document. In many cases, however, observing the system calls made by a process can reveal a great deal about what it is doing. This can be accomplished using the `truss` utility on Solaris or the `strace` tool on Linux.

The `truss` utility is very powerful and has a lot of options, but two of the most useful forms in which it may be invoked are:

- **truss -f -p {processID}** – Provides a basic overview of all system calls being made by the specified process (and any subprocesses that it creates) and their associated return values.
- **truss -fear all -p {processID}** – Provides an extremely verbose trace of all system call activity, including details about data being read from or written to files and sockets.

In both cases, the output may be written to a file instead of the terminal window by adding the `-o {path}` option. Further, rather than observing an already-running process, it is possible to have `truss` launch the process and trace execution over its entire life span by replacing `-p {processID}` with name and arguments for the command to invoke.

On Linux systems, the basic equivalent of the first truss variant above is:

```
strace -f -p {processID}
```

Consult the `strace` manual page for additional information about using it to trace process execution on Linux.

Examining Network Communication

Because the UnboundID® Directory Proxy Server is a network-based application, it can be valuable to observe the network communication that it has with clients. The Directory Proxy Server itself can provide details about its interaction with clients by enabling debugging for the protocol or data debug categories, but there may be a number of cases in which it is useful to view information at a much lower level. A network sniffer, like the `snoop` tool on Solaris or 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 commands can be used on Solaris and Linux, respectively:

```
snoop -d {interface} -r -x 0 host 1.2.3.4 port 389
tcpdump -i {interface} -n -XX -s 0 host 1.2.3.4 and port 389
```

On Solaris systems, the `snoop` command provides enhanced support for parsing LDAP communication (but only when the Directory Proxy Server is listening on the default port of 389). By adding the "`-v`" argument to the `snoop` command line, a verbose breakdown of each packet will be displayed, including protocol-level information. It does not appear that the `tcpdump` tool provides support for LDAP parsing. However, in either case it is possible to write capture data to a file rather than displaying information on the terminal (using "`-o {path}`" with `snoop`, or "`-w {path}`" with `tcpdump`), so that information can be later analyzed with a graphical tool like Wireshark, which provides the ability to interpret LDAP communication on any port.

Note that enabling network tracing generally requires privileges that are not available to normal users and therefore may require root access. On Solaris systems, granting the `net_rawaccess` privilege to a user should be sufficient to allow that user to run the `snoop` utility.

Common Problems and Potential Solutions

This section describes a number of different types of problems that can occur and common potential causes for them.

General Methodology to Troubleshoot a Problem

When a problem is detected, UnboundID recommends using the following general methodology to isolate the problem:

1. Run the `bin/status` tool or look at the server status in the web 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 bottlenecked 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 directory service 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 UnboundID® Directory Proxy Server requires that Java 1.6 (minimum version: Sun/Oracle JDK 1.6.0_31 or JRE 1.6.0 IBM J9 2.4) be installed on the system and made available to the server, and it must be installed prior to running `setup`. If the `setup` tool does not detect that a suitable Java environment is available, it will refuse to run.

To ensure that this does not happen, the `setup` tool should be invoked with an explicitly-defined value for the *JAVA_HOME* environment variable that specifies the path to the Java installation that should be used. For example:

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

If this still does not work for some reason, then it can be that the value specified in the provided *JAVA_HOME* environment variable can be overridden by another environment variable. If that occurs, try the following command, which should override any other environment variables that can be set:

```
env UNBOUNDID_JAVA_HOME="/ds/java" UNBOUNDID_JAVA_BIN="" ./setup
```

Unexpected Arguments Provided to the JVM

If the `setup` script attempts to launch the java command with an invalid set of Java arguments, it might prevent the JVM from starting. By default, no special options are provided to the JVM when running setup, but this might not be the case if either the *JAVA_ARGS* or *UNBOUNDID_JAVA_ARGS* environment variable is set. If the `setup` tool displays an error message that indicates that the Java environment could not be started with the provided set of arguments, then invoke the following command before trying to re-run `setup`:

```
unset JAVA_ARGS UNBOUNDID_JAVA_ARGS
```

The Server Has Already Been Configured or Used

The `setup` tool is only intended to provide the initial configuration for the Directory Proxy Server. It refuses to run if it detects that the `setup` tool has already been run, or if an attempt has been made to start the Directory Proxy Server prior to running the `setup` tool. This protects an existing Directory Proxy Server installation from being inadvertently updated in a manner that could harm an existing configuration or data set.

If the Directory Proxy Server has been previously used and if you want to perform a fresh installation, it is recommended that you first remove the existing installation, create a new one and run `setup` in that new installation. However, if you are confident that there is nothing of value in the existing installation (for example, if a previous attempt to run `setup` failed to

complete successfully for some reason but it will refuse to run again), the following steps can be used to allow the `setup` program to run:

- Remove the `config/config.ldif` file and replace it with the `config/update/config.ldif.{revision}` file containing the initial configuration.
- If there are any files or subdirectories below the `db` directory, then remove them.
- If a `config/java.properties` file exists, then remove it.
- If a `lib/setup-java-home` script (or `lib\set-java-home.bat` file on Microsoft Windows) exists, then remove it.

The Server Will Not Start

If the Directory Proxy Server does not start, then there are a number of potential causes.

The Server or Other Administrative Tool Is Already Running

Only a single instance of the Directory Proxy Server can run at any time from the same installation root. If an instance is already running, then subsequent attempts to start the server will fail. Similarly, some other administrative operations can also prevent the server from being started. In such cases, the attempt to start the server should fail with a message like:

```
The Directory Proxy Server could not acquire an exclusive lock on file
/ds/UnboundID-Proxy/locks/server.lock: The exclusive lock requested for file
/ds/UnboundID-Proxy/locks/ server.lock was not granted, which indicates
that another process already holds a shared or exclusive lock on that
file. This generally means that another instance of this server is already
running
```

If the Directory Proxy Server is not running (and is not in the process of starting up or shutting down) and there are no other tools running that could prevent the server from being started, and the server still believes that it is running, then it is possible that a previously-held lock was not properly released. In that case, you can try removing all of the files in the `locks` directory before attempting to start the server.

If you wish to have multiple instances running at the same time on the same system, then you should create a completely separate installation in another location on the filesystem.

There Is Not Enough Memory Available

When the Directory Proxy Server is started, the JVM attempts to allocate all memory that it has been configured to use. If there is not enough free memory available on the system, then the Directory Proxy Server generates an error message that indicates that the server could not be started with the specified set of arguments. Note that it is possible that an invalid option was provided to the JVM (as described below), but if that same set of JVM arguments has already been used successfully to run the server, then it is more likely that the system does not have enough memory available.

There are a number of potential causes for this:

- If the amount of memory in the underlying system has changed (for example, system memory has been removed, or if the Directory Proxy Server is running in a zone or other type of virtualized container and a change has been made to the amount of memory that container will be allowed to use), then the Directory Proxy Server might need to be re-configured to use a smaller amount of memory than had been previously configured.
- Another process running on the system is consuming a significant amount of memory so that there is not enough free memory available to start the server. If this is the case, then either terminate the other process to make more memory available for the Directory Proxy Server, or reconfigure the Directory Proxy Server to reduce the amount of memory that it attempts to use.
- The Directory Proxy Server was just shut down and an attempt was made to immediately restart it. In some cases, if the server is configured to use a significant amount of memory, then it can take a few seconds for all of the memory that had been in use by the server, when it was previously running, to be released back to the operating system. In that case, run the `vmstat` command and wait until the amount of free memory stops growing before attempting to restart the server.
- For Solaris-based systems only, if the system has one or more ZFS filesystems (even if the Directory Proxy Server itself is not installed on a ZFS filesystem), but it has not been configured to limit the amount of memory that ZFS can use for caching, then it is possible that ZFS caching is holding onto a significant amount of memory and cannot release it quickly enough when it is needed by the Directory Proxy Server. In that case, the system should be re-configured to limit the amount of memory that ZFS is allowed to use as described in the Using the Collect Support Data Tool.
- If the system is configured with one or more memory-backed filesystems, for example, `tmpfs` used for `/tmp` for Solaris), then look to see if there are any large files that can be consuming a significant amount of memory in any of those locations. If so, then remove them or relocate them to a disk-based filesystem.
- For Linux systems only, if there is a mismatch between the huge pages setting for the JVM and the huge pages reserved in the operating system. For more information, see Configure Huge Page Support (Linux).

If nothing else works and there is still not enough free memory to allow the JVM to start, then as a last resort, try rebooting the system.

An Invalid Java Environment or JVM Option Was Used

If an attempt to start the Directory Proxy Server fails with an error message indicating that no valid Java environment could be found, or indicates that the Java environment could not be started with the configured set of options, then you should first ensure that enough memory is available on the system as described above. If there is a sufficient amount of memory available, then other causes for this error can include the following:

- The Java installation that was previously used to run the server no longer exists (for example, an updated Java environment was installed and the old installation was removed). In that case, update the `config/java.properties` file to reference to path to the new Java installation and run the `bin/dsjavaproperties` command to apply that change.

- The Java installation used to run the server has been updated and the server is trying to use the correct Java installation but one or more of the options that had worked with the previous Java version no longer work with the new version. In that case, it is recommended that the server be re-configured to use the previous Java version, so that it can be run while investigating which options should be used with the new installation.
- If an *UNBOUNDID_JAVA_HOME* or *UNBOUNDID_JAVA_BIN* environment variable is set, then its value may override the path to the Java installation used to run the server as defined in the `config/java.properties` file. Similarly, if an *UNBOUNDID_JAVA_ARGS* environment variable is set, then its value might override the arguments provided to the JVM. If this is the case, then explicitly unset the *UNBOUNDID_JAVA_HOME*, *UNBOUNDID_JAVA_BIN*, and *UNBOUNDID_JAVA_ARGS* environment variables before trying to start the server.

Note that any time the `config/java.properties` file is updated, the `bin/dsjavaproperties` tool must be run to apply the new configuration. If a problem with the previous Java configuration prevents the `bin/dsjavaproperties` tool from running properly, then it can be necessary to remove the `lib/set-java-home` script (or `lib\set-java-home.bat` file on Microsoft Windows) and invoke the `bin/dsjavaproperties` tool with an explicitly-defined path to the Java environment, like:

```
env UNBOUNDID_JAVA_HOME=/ds/java bin/dsjavaproperties
```

An Invalid Command-Line Option Was Provided

There are a small number of arguments that are provided when running the `bin/start-ds` command, but in most cases, none are required. If one or more command-line arguments were provided for the `bin/start-ds` command and any of them is not recognized, then the server provides an error message indicating that an argument was not recognized and displays version information. In that case, correct or remove the invalid argument and try to start the server again.

The Server Has an Invalid Configuration

If a change is made to the Directory Proxy Server configuration using an officially-supported tool like `dsconfig` or the directory management console, the server should validate that configuration change before applying it. However, it is possible that a configuration change can appear to be valid at the time that it is applied, but does not work as expected when the server is restarted. Alternately, a change in the underlying system can cause a previously-valid configuration to become invalid.

In most cases involving an invalid configuration, the Directory Proxy Server displays (and writes to the error log) a message that explains the problem, and this can be sufficient to identify the problem and understand what action needs to be taken to correct it. If for some reason the startup failure does not provide enough information to identify the problem with the configuration, then look in the `logs/config-audit.log` file to see what recent configuration changes have been made with the server online, or in the `config/archived-configs` directory to see if there might have been a recent configuration change resulting from a direct change to the configuration file itself that was not made through a supported configuration interface.

If the server does not start as a result of a recent invalid configuration change, then it can be possible to start the server using the configuration that was in place the last time that the server started successfully (for example, the "last known good" configuration). This can be achieved using the `--useLastKnownGoodConfig` option:

```
$ bin/start-ds --useLastKnownGoodConfig
```

Note that if it has been a long time since the last time the server was started and a number of configuration changes have been made since that time, then the last known good configuration can be significantly out of date. In such cases, it can be preferable to manually repair the configuration.

If there is no last known good configuration, if the server no longer starts with the last known good configuration, or if the last known good configuration is significantly out of date, then manually update the configuration by editing the `config/config.ldif` file. In that case, you should make sure that the server is offline and that you have made a copy of the existing configuration before beginning. You might wish to discuss the change with your authorized support representative before applying it to ensure that you understand the correct change that needs to be made.

Note: In addition to manually-editing the config file, you can look at previous achived configurations to see if the most recent one works. You can also use the `ldif-diff` tool to compare the configurations in the archive to the current configuration to see what is different.

You Do Not Have Sufficient Permissions

The Directory Proxy Server should only be started by the user or role used to initially install the server. In most cases, if an attempt is made to start the server as a user or role other than the one used to create the initial configuration, then the server will fail to start, because the user will not have sufficient permissions to access files owned by the other user, such as database and log files. However, if the server was initially installed as a non-root user and then the server is started by the root account, then it can no longer be possible to start the server as a non-root user because new files that are created would be owned by root and could not be written by other users.

If the server was inadvertently started by root when it is intended to be run by a non-root user, or if you wish to change the user account that should be used to run the server, then it should be sufficient to simply change ownership on all files in the Directory Proxy Server installation, so that they are owned by the user or role under which the server should run. For example, if the Directory Proxy Server should be run as the "ds" user in the "other" group, then the following command can be used to accomplish this (invoked by the root user):

```
chown -R ds:other /ds/UnboundID-Proxy
```

The Server Has Crashed or Shut Itself Down

You can first check the current server state by using the `bin/server-state` command. If the Directory Proxy Server was previously running but is no longer active, then the potential reasons include the following:

- The Directory Proxy Server was shut down by an administrator. Unless the server was forcefully terminated (for example, using “kill -9”), then messages are written to the `error` and `server.out` logs explaining the reason for the shutdown.
- The Directory Proxy Server was shut down when the underlying system crashed or was rebooted. If this is the case, then running the `uptime` command on the underlying system shows that it was recently booted.
- The Directory Proxy Server process was terminated by the underlying operating system for some reason (for example, the out of memory killer on Linux). If this happens, then a message will be written to the system error log.
- The Directory Proxy Server decided to shut itself down in response to a serious problem that had arisen. At present, this should only occur if the server has detected that the amount of usable disk space has become critically low, or if significant errors have been encountered during processing that left the server without any remaining worker threads to process operations. If this happens, then messages are written to the `error` and `server.out` logs (if disk space is available) to provide the reason for the shutdown.
- The JVM in which the Directory Proxy Server was running crashed. If this happens, then the JVM should dump a fatal error log (a `hs_err_pid{processID}.log` file) and potentially a core file.

In the event that the operating system itself crashed or terminated the process, then you should work with your operating system vendor to diagnose the underlying problem. If the JVM crashed or the server shut itself down for a reason that is not clear, then contact your authorized support provider for further assistance.

The Server Will Not Accept Client Connections

You can first check the current server state by using the `bin/server-state` command. If the Directory Proxy Server does not appear to be accepting connections from clients, then potential reasons include the following:

- The Directory Proxy Server is not running.
- The underlying system on which the Directory Proxy Server is installed is not running.
- The Directory Proxy Server is running but is not reachable as a result of a network or firewall configuration problem. If that is the case, then connection attempts should time out rather than be rejected.
- If the Directory Proxy Server is configured to allow secure communication via SSL or StartTLS, then a problem with the key manager and/or trust manager configuration can cause

connections to be rejected. If that is the case, then messages should be written to the server access log for each failed connection attempt.

- If the Directory Proxy Server has been configured with a maximum allowed number of connections, then it can be that the maximum number of allowed client connections are already established. If that is the case, then messages should be written to the server access log for each rejected connection attempt.
- If the Directory Proxy Server is configured to restrict access based on the address of the client, then messages should be written to the server access log for each rejected connection attempt.
- If a connection handler encounters a significant error, then it can stop listening for new requests. If this occurs, then a message should be written to the server error log with information about the problem. Another solution is to restart the server. A third option is to restart the connection handler using the LDIF connection handler to make it available again. To do this, create an LDIF file that disables and then re-enables the connection handler, create the `config/auto-process-ldif` directory if it does not already exist, and then copy the LDIF file into it.

The Server is Unresponsive

You can first check the current server state by using the `bin/server-state` command. If the Directory Proxy Server process is running and appears to be accepting connections but does not respond to requests received on those connections, then potential reasons for this behavior include:

- If all worker threads are busy processing other client requests, then new requests that arrive will be forced to wait in the work queue until a worker thread becomes available. If this is the case, then a stack trace obtained using the `jstack` command shows that all of the worker threads are busy and none of them are waiting for new requests to process.

Note: If all of the worker threads are tied up processing the same operation for a long time, the server will also issue an alert that it might be deadlocked, which may not actually be the case. All threads might be tied up processing unindexed searches.

- If a request handler is stuck performing some expensive processing for a client connection, then other requests sent to the server on connections associated with that request handler is forced to wait until the request handler is able to read data on those connections. If this is the case, then only some of the connections can experience this behavior (unless there is only a single request handler, in which it will impact all connections), and stack traces obtained using the `jstack` command shows that a request handler thread is continuously blocked rather than waiting for new requests to arrive. Note that this scenario is a theoretical problem and one that has not appeared in production.
- If the JVM in which the Directory Proxy Server is running is not properly configured, then it can be forced to spend a significant length of time performing garbage collection, and in severe cases, could cause significant interruptions in the execution of Java code. In such cases, a stack trace obtained from a `pstack` of the native process should show that most

threads are idle but at least one thread performing garbage collection is active. It is also likely that one or a small number of CPUs is 100% busy while all other CPUs are mostly idle. The server will also issue an alert after detecting a long JVM pause (due to garbage collection). The alert will include details of the pause.

- If the JVM in which the Directory Proxy Server is running has hung for some reason, then the `pstack` utility should show that one or more threads are blocked and unable to make progress. In such cases, the system CPUs should be mostly idle.
- If a network or firewall configuration problem arises, then attempts to communicate with the server cannot be received by the server. In that case, a network sniffer like `snoop` or `tcpdump` should show that packets sent to the system on which the Directory Proxy Server is running are not receiving TCP acknowledgement.
- If the system on which the Directory Proxy Server is running has become hung or lost power with a graceful shutdown, then the behavior is often similar to that of a network or firewall configuration problem.

If it appears that the problem is with the Directory Proxy Server software or the JVM in which it is running, then you need to work with your authorized support provider to fully diagnose the problem and determine the best course of action to correct it.

The Server is Slow to Respond to Client Requests

If the Directory Proxy Server is running and does respond to clients, but clients take a long time to receive responses, then the problem can be attributable to a number of potential problems. In these cases, use the Periodic Stats Logger, which is a valuable tool to get per-second monitoring information on the Directory Proxy Server. The Periodic Stats Logger can save the information in csv format for easy viewing in a spreadsheet. For more information, see "Profiling Server Performance Using the Periodic Stats Logger". The potential problems that cause slow responses to client requests are as follows:

- The server is not optimally configured for the type of requests being processed, or clients are requesting inefficient operations. If this is the case, then the access log should show that operations are taking a long time to complete and they will likely be unindexed. In that case, updating the server configuration to better suit the requests, or altering the requests to make them more efficient, could help alleviate the problem. In this case, view the expensive operations access log in `logs/expensive-ops`, which by default logs operations that take longer than 1 second. You can also run the `bin/status` command or view the status in the web console to see the Directory Proxy Server's Work Queue information (also see the next bullet point).
- The server is overwhelmed with client requests and has amassed a large backlog of requests in the work queue. This can be the result of a configuration problem (for example, too few worker thread configured), or it can be necessary to provision more systems on which to run the Directory Proxy Server software. Symptoms of this problem appear similar to those experienced when the server is asked to process inefficient requests, but looking at the details of the requests in the access log show that they are not necessarily inefficient requests. Run the `bin/status` command to view the Work Queue information. If everything is performing well, you should not see a large queue size or a server that is near 100% busy. The %Busy statistic is calculated as the percentage of worker threads that are busy processing operations.

```
           --- Work Queue ---
           : Recent : Average : Maximum
-----------:--------:---------:--------
Queue Size : 10  : 1       : 10
% Busy     : 17     : 14      : 100
```

You can also view the expensive operations access log in `logs/expensive-ops`, which by default logs operations that take longer than 1 second.

- The server is not configured to fully cache all of the data in the server, or the cache is not yet primed. In this case, `iostat` reports a very high disk utilization. This can be resolved by configuring the server to fully cache all data, and to load database contents into memory on startup. If the underlying system does not have enough memory to fully cache the entire data set, then it might not be possible to achieve optimal performance for operations that need data which is not contained in the cache. For more information, see Disk-Bound Deployments.
- If the JVM is not properly configured, then it will need to perform frequent garbage collection and periodically pause execution of the Java code that it is running. In that case, the server error log should report that the server has detected a number of pauses and can include tuning recommendations to help alleviate the problem.
- If the Directory Proxy Server is configured to use a large percentage of the memory in the system, then it is possible that the system has gotten low on available memory and has begun swapping. In this case, `iostat` should report very high utilization for disks used to hold swap space, and commands like `swap -l` on Solaris or `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 (Linux).
- If another process on the system is consuming a significant amount of CPU time, then it can adversely impact the ability of the Directory Proxy Server to process requests efficiently. Isolating the processes (for example, using processor sets) or separating them onto different systems can help eliminate this problem.

The Server Returns Error Responses to Client Requests

If a large number of client requests are receiving error responses, then view the `logs/failed-ops` log, which is an access log for only failed operations. The potential reasons for the error responses include the following:

- If clients are requesting operations that legitimately should fail (for example, they are targeting entries that do not exist, are attempting to update entries in a way that would violate the server schema, or are performing some other type of inappropriate operation), then the problem is likely with the client and not the server.
- If a portion of the Directory Proxy Server data is unavailable (for example, because an online LDIF import or restore is in progress), then operations targeting that data will fail. Those problems will be resolved when the backend containing that data is brought back online. During the outage, it might be desirable to update proxy servers or load balancers or both to route requests away from the affected server. As of Directory Proxy Server version 3.1 or later, the Directory Proxy Server will indicate that it is in a degraded status and the proxy server will route around it.

- If the Directory Proxy Server work queue is configured with a maximum capacity and that capacity has been reached, then the server begins rejecting all new requests until space is available in the work queue. In this case, it might be necessary to alter the server configuration or the client requests or both, so that they can be processed more efficiently, or it might be necessary to add additional server instances to handle some of the workload.
- If an internal error occurs within the server while processing a client request, then the server terminates the connection to the client and logs a message about the problem that occurred. This should not happen under normal circumstances, so you will need to work with your authorized support provider to diagnose and correct the problem.
- If a problem is encountered while interacting with the underlying database (for example, an attempt to read from or write to disk failed because of a disk problem or lack of available disk space), then it can begin returning errors for all attempts to interact with the database until the backend is closed and re-opened and the database has been given a change to recover itself. In these cases, the `je.info.*` file in the database directory should provide information about the nature of the problem.

Problems with the Directory Management Console

If a problem arises when trying to use the directory management console, then potential reasons for the problem may include the following:

- The web application container used to host the console is not running. If an error occurs while trying to start it, then consult the logs for the web application container.
- If a problem occurs while trying to authenticate to the web application container, then make sure that the target Directory Proxy Server is online. If it is online, then the access log may provide information about the reasons for the authentication failure.
- If a problem occurs while attempting to interact with a Directory Proxy Server instance using the Directory Proxy Management Console, then the access and error logs for that Directory Proxy Server instance might provide additional information about the underlying problem.

Problems with the Directory Management Console: JVM Memory Issues

Console runs out of memory (PermGen). If you are running a Management Console for a UnboundID® Directory Server while also running a console for the UnboundID® Directory Proxy Server Management Console and an UnboundID® Synchronization Server Management Console, you may see a Java PermGen error as follows:

```
Exception in thread "http-bio-8080-exec-7" java.lang.OutOfMemoryError: PermGen Space
```

For a servlet container, such as Tomcat, you can specify additional arguments to pass to the JVM by creating a `bin/setenv.sh` file (or `setenv.bat` for Windows) that sets the CATALINA_OPTS variable. The `startup.sh` script will automatically pick this up. For example:

```
#!/bin/bash
# The following may be modified to change JVM memory arguments.
MAX_HEAP_SIZE=512m
MIN_HEAP_SIZE=$MAX_HEAP_SIZE
```

```
MAX_PERM_SIZE=256m

CATALINA_OPTS="-Xmx${MAX_HEAP_SIZE} -Xms${MIN_HEAP_SIZE} -XX:MaxPermSize=
${MAX_PERM_SIZE}"
```

Global Index Growing Too Large

If the global index appears to be growing too large, you can reload from the backend directory servers. Use the `reload-index` tool with the `--fromDS` option, overriding the configuration of the `prime-index-source` property. You can do this on a one off basis if the global index appears to be growing too large as follows:

```
$ bin/reload-index \
  --bindPassword password \
  --baseDN "dc=example,dc=com" \
  --fromDS
```

Forgotten Proxy User Password

If you have forgotten the password you set for the `cn=Proxy User` entry, you can work around the problem as follows:

- You can temporarily add a second password to the proxy user entry so that you can transition all of the directory proxy server instances to the new password. However, you should have multiple passwords on the `cn=Proxy User` entry for the shortest time possible.
- If you do not know the clear-text value, then you can use the encrypted value when configuring the new Directory Proxy Server. The encryption scheme allows reversible passwords that are stored in the server configuration so that they can be decrypted by any server instance.
- You can create a new root user in the directory server instances with the appropriate set of privileges and have the new directory proxy server instance use that account to authenticate. Since it is not a good idea to have an account for which you do not know the password, you may want to update all of the other directory proxy server instances to use the new account.
- You can use a protocol analyzer like `snoop` or `Wireshark`, or a tool like the `LDAPDecoder` provided with SLAMD, to capture the password from the network communication. When used in proxy mode, the SLAMD `LDAPDecoder` tool can handle SSL-encrypted communication, and any of these methods can handle clear-text LDAP.

Providing Information for Support Cases

If a problem arises that you are unable to fully diagnose and correct on your own, then contact your authorized support provider for assistance. To ensure that the problem can be addressed as quickly as possible, be sure to provide all of the information that the support personnel may need to fully understand the underlying cause by running the `collect-support-data` tool, and then sending the generated zip file to your authorized support provider. It is good practice to run this tool and send the ZIP file to your authorized support provider before any corrective action has taken place.

Chapter 9 Managing Extensions

The UnboundID® Directory Proxy Server provides support for any custom extensions that you create using the Server SDK as well as the Simple Cloud Identity Management (SCIM) protocol extension. This chapter presents information on how to install and work with these extensions.

Topics:

Managing Directory Proxy Server Extensions

You can create extensions that use the Server SDK to extend the functionality of your Directory Proxy Server. Extension bundles are installed from a .zip archive or a file system directory. You can use the `manage-extension` tool to install or update any extension that is packaged using the extension bundle format. It opens and loads the extension bundle, confirms the correct extension to install, stops the server if necessary, copies the bundle to the server install root, and then restarts the server.

Note: The `manage-extension` tool may only be used with Java extensions packaged using the extension bundle format. Groovy extensions do not use the extension bundle format. For more information, see the "Building and Deploying Java-Based Extensions" section of the Server SDK documentation, which describes the extension bundle format and how to build an extension.

Introduction to the SCIM Servlet Extension

The SCIM servlet extension works with the UnboundID® Directory Proxy Server to facilitate moving users to, from, and between clouds in a secure, fast, and simple way. This section describes fundamental SCIM concepts and provides information about creating your own SCIM application.

Overview of SCIM Fundamentals

Understanding the basic concepts of SCIM will help you understand how to use the SCIM extension to meet the needs of your deployment. SCIM allows you to:

- **Provision identities.** Through the API, you have access to the basic create, read, update, and delete functions, as well as other special functions.
- **Provision groups.** SCIM also allows you to manage groups.
- **Interoperate using a common schema.** SCIM provides a well-defined, platform-neutral user and group schema, as well as a simple mechanism to extend it.

The SCIM extension supports the 1.1 implementation 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 at: http://www.simplecloud.info/.

Before You Begin

The process for setting up a SCIM extension involves setting up the SCIM extension using either the `scim-config-ds.dsconfig` file for the Directory Server or `scim-config-proxy.dsconfig` file for the Directory Proxy Server. Both files are located in the `config` directory for each respective server and can be run as a `dsconfig` batch process.

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.

Configuring the SCIM Servlet Extension

A default SCIM HTTP Servlet Extension is provided out of the box with the Directory Proxy Server. You can run a `dsconfig` batch script, located in the `config` directory. The script runs a series of commands that enables the SCIM HTTP Connection Handler, increases the level of detail logged by the HTTP Detailed Access log publisher, and adds access controls to allow access to LDAP controls used by the SCIM Servlet Extension. There are additional optional configurations (e.g., changing the log format and using VLV indexes) that you can make by altering the `dsconfig` batch scripts.

1. 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, open the `scim-config-ds.dsconfig` file (Directory Server) or `scim-config-proxy.dsconfig` (Proxy Server), and remove the comment (#) symbol on the following `dsconfig` commands. Save the file when finished editing:

```
# The SCIM HTTP Connection Handler automatically uses a detailed HTTP log publisher
# which uses an UnboundID proprietary format. If you need a standard W3C common
# log format publisher instead, use the following commands:
#
# 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"
#
# dsconfig set-connection-handler-prop --handler-name "SCIM HTTP Connection Handler"
 --add "http-operation-log-publisher:HTTP Common Access"
```

2. If you want to support pagination, create some Virtual List View indexes. open the `scim-config-ds.dsconfig` file (Directory Server) or `scim-config-proxy.dsconfig` (Proxy Server), and remove the comment (#) symbol on the following `dsconfig` and `rebuild-index` commands. Save the file when finished editing:

```
#
# Optionally, create some Virtual List View indexes to support pagination.
#

# 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"
```

```
# 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"

#
# Command to build the Virtual List View indexes just created:
#
# rebuild-index --baseDN dc=example,dc=com --index vlv.ascending-uid --index
 vlv.ascending-sn
#
```

3. To enable the SCIM servlet extension, run the `dsconfig` batch file.

```
$ bin/dsconfig --batch-file config/scim-config-ds.dsconfig
```

Verifying the SCIM Servlet Extension Configuration

You can verify the configuration of the SCIM extension by navigating to a SCIM URL via the command line or through a browser window. For example, `curl` is used in the following command-line example to verify that the SCIM plugin is running. The `-k` (or `--insecure`) option is used to turn off curl's verification of the server certificate, since the example Directory Server is using a self-signed certificate.

```
$ curl -u "cn=Directory Manager:password" \
-k "https://localhost:8443/ServiceProviderConfigs"

{"schemas":["urn:scim:schemas:core:1.0"],"id":"urn:scim:schemas:core:1.0",
"patch":{"supported":true},"bulk":{"supported":true,"maxOperations":10000,
"maxPayloadSize":10485760},"filter":{"supported":true,"maxResults":100},
"changePassword":{"supported":true},"sort":{"supported":true},
"etag":{"supported":false},"authenticationSchemes":[{"name":"HttpBasic",
"description":"The HTTP Basic Access Authentication scheme. This scheme is
not considered to be a secure method of user authentication (unless used in
conjunction with some external secure system such as SSL), as the user
name and password are passed over the network as cleartext.","specUrl":
"http://www.ietf.org/rfc/rfc2617","documentationUrl":
"http://en.wikipedia.org/wiki/Basic_access_authentication"}]}
```

If the user ID is a valid DN (such as `cn=Directory Manager`), the SCIM extension authenticates by binding to the Directory 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 directory as that user. The following command line verifies authentication to the directory as the user with the uid of user.0.

```
$ curl -u "user.0:password" \
  -k "https://localhost:8443/ServiceProviderConfigs"
```

About the UnboundID SCIM Implementation

This section discusses details about the UnboundID implementation of the SCIM protocol. Before reading this chapter, familiarize yourself with the SCIM Protocol specification, available from http://www.simplecloud.info/.

Summary of SCIM Protocol Support

UnboundID implements all required features of the SCIM protocol and most optional features. The following table describes SCIM features and whether they are supported by UnboundID.

Table 10: SCIM Protocol Support

SCIM Feature	Supported
JSON	Yes
XML*	Yes
Authentication/Authorization	Yes, via HTTP basic authentication or OAuth 2.0 bearer tokens
Service Provider Configuration	Yes
Schema	Yes
User resources	Yes
Group resources	Yes
User-defined resources	Yes
Resource retrieval via GET	Yes
List/query resources	Yes
Query filtering*	Yes
Query result sorting*	Yes
Query result pagination*	Yes
Resource updates via PUT	Yes
Partial resource updates via PATCH*	Yes
Resource deletes via DELETE	Yes
Resource versioning*	No
Bulk*	Yes
HTTP method overloading	Yes

* denotes an optional feature of the SCIM protocol.

About the HTTP Log Publisher

HTTP operations may be logged using either a Common Log File HTTP Operation Log Publisher or a Detailed HTTP Operation Log Publisher. The Common Log File HTTP Operation Log Publisher is a built-in log publisher that records HTTP operation information to a file using the W3C common log format. Because the W3C common log format is used, logs produced by this log publisher can be parsed by many existing web analysis tools.

Log messages are formatted as follows:

- IP address of the client.
- RFC 1413 identification protocol. The Ident Protocol is used to format information about the client.

- The user ID provided by the client in an Authorization header, which is typically available server-side in the REMOTE_USER environment variable. A dash appears in this field if this information is not available.
- A timestamp, formatted as "'['dd/MM/yyyy:HH:mm:ss Z']'"
- Request information, with the HTTP method followed by the request path and HTTP protocol version.
- The HTTP status code value.
- The content size of the response body in bytes. This number does not include the size of the response headers.

The HTTP Detailed Access Log Publisher provides more information that 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 generated log messages such as the following. The lines have been wrapped for readability.

```
[15/Feb/2012:21:17:04 -0600] RESULT requestID=10834128
from="10.2.1.114:57555" method="PUT"
url="https://10.2.1.129:443/Aleph/Users/6272c691-
38c6-012f-d227-0dfae261c79e" authorizationType="Basic"
requestContentType="application/json" statusCode=200
etime=3.544 responseContentLength=1063
redirectURI="https://server1.example.com:443/Aleph/Users/6272c691-38c6-012f-
d227-0dfae261c79e"
responseContentType="application/json"
```

In this example, only default log publisher properties are used. Though this message is for a RESULT, it contains information about the request, such as the client address, the request method, the request URL, the authentication method used, and the Content-Type requested. For the response, it includes the response length, the redirect URI, the Content-Type, and the HTTP status code.

You can modify the information logged, including adding request parameters, cookies, and specific request and response headers. For more information, refer to the `dsconfig` command-line tool help.

Configuring ACIs for the SCIM Servlet Extension

The SCIM extension configuration file modifies ACIs so that the Directory Proxy Server can receive updates through SCIM. For example, without the ACI modifications in the `scim-config-ds.dsconfig` (Directory only) or `scim-config-proxy.dsconfig` (Proxy only), only `cn=Directory Manager` can perform updates via SCIM. The `scim-config-ds.dsconfig` (Directory only) or `scim-config-proxy.dsconfig` (Proxy only) contain all the ACI's needed and can be run as a `dsconfig` batch script. The remainder of this section describes how to add the access controls needed by the SCIM Servlet Extension.

To Add ACIs to Allow LDAP Control Access

You need to add access controls to allow access to the LDAP controls used by the SCIM Servlet Extension. These controls are the Post-Read Request Control (1.3.6.1.1.13.2), the Server-Side Sort Request Control (1.2.840.113556.1.4.473) and the Virtual List View Request Control (2.16.840.1.113730.3.4.9).

- Add the controls using `dsconfig` as follows:

```
$ 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 || 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";)'
```

To Add ACIs to Allow Read Access to Operational Attributes

You also need to add access controls that allow read access to the operational attributes used by the SCIM Servlet Extension.

- Add these controls as follows:

```
$ bin/dsconfig set-access-control-handler-prop \
  --add 'global-aci:(targetattr="entryUUID || \
    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";)'
```

Configuring VLV Indexes (Directory Server Only)

You need to create some Virtual List View indexes so that SCIM can support pagination. For example, you might add the following VLV indexes using `dsconfig`:

```
$ 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"
```

After you have created the VLV indexes, you need to build them using the following command:

```
$ bin/rebuild-index --baseDN dc=example,dc=com --index vlv.ascending-uid \
  --index vlv.ascending-sn
```

Configuring LDAP Control Support on All Request Processors (Proxy Only)

You need to configure support for the required LDAP controls on all request processors handling LDAP requests that result from SCIM requests. Change the request processor name that was provided as an example and repeat the command for all additional request processors.

```
$ bin/dsconfig set-request-processor-prop \
  --processor-name dc_example_dc_com-req-processor \
  --add supported-control-oid:1.2.840.113556.1.4.473 \
  --add supported-control-oid:2.16.840.1.113730.3.4.9
```

Bulk Operation Implementation

The SCIM extension supports bulk operations as specified in the SCIM protocol. The remainder of this section describes details about how bulk operations are executed in a bulk request, memory and disk usage, and status codes.

BulkId References

Bulk operations within a bulk request are executed in the order that the client presents the operations. So, any forward `bulkId` references will result in an error. For example, if a bulk request creates a new user using a POST and assigns that user to a group using a PUT, then the POST must appear before the PUT.

In addition to allowing `bulkId` references in resource data, the UnboundID implementation allows a `bulkId` reference in the path of a bulk operation. For example, the client could POST a new user with a `bulkId` of `user1`. Then later, in the same bulk request, the client could PUT that same user using the path `/Users/bulkId:user1`.

Memory and Disk Usage

The SCIM extension tries to minimize the amount of heap memory required to process a bulk request by writing temporary data to files in the `tmpDataDir` directory. A significant amount of heap memory may still be used to resolve `bulkId` references to resource IDs. The amount of heap memory needed is bounded by the maximum size of a bulk request (`bulkMaxPayloadSize`) multiplied by the maximum number of concurrent bulk requests (`bulkMaxConcurrentRequests`). Typically, the actual amount of memory used is far less.

Care should be taken to ensure that the Directory Proxy Server running the SCIM extension has enough total heap to allow for the memory needed by the SCIM extension. If necessary, reduce the `bulkMaxPayloadSize` and/or `bulkMaxConcurrentRequests` settings.

Overview of Status Codes

The most common status codes that may be returned by a bulk request follow:

- **200** - The bulk request was processed, although one or more of the contained operations may have failed or may not have been valid.
- **400** - The bulk request could not be understood.
- **413** - The request exceeds the `bulkMaxOperations` or `bulkMaxPayloadSize` limits.
- **503** - The `bulkMaxConcurrentRequests` limit would have been exceeded.

Mapping SCIM Resource IDs

The default `scim-resources.xml` configuration maps the SCIM resource ID to the LDAP entry DN. However, the entry DN does not necessarily meet the requirements of the SCIM specification regarding resource ID immutability. LDAP permits entries to be renamed or moved, thus modifying the DN (the DN can only be changed through the LDAP interface, not through the SCIM interface). The resource configuration allows the SCIM resource ID to be mapped to an LDAP attribute as an alternative to mapping to the LDAP entry DN. The `entryUUID` attribute, whose read-only value is assigned by the Directory Server, is a possible alternative mapping. However, configuring a mapping to an attribute, rather than the entry DN, may result in inefficient group processing, since LDAP groups use the entry DN as the basis of group membership.

A resource may also be configured such that its SCIM resource ID is provided by an arbitrary attribute in the request body during POST operations. This SCIM attribute must be mapped to an LDAP attribute so that the SCIM resource ID may be stored in the Directory Server. By default, it is the responsibility of the SCIM client to guarantee ID uniqueness. However, the UID Unique Attribute Plugin may be used by the Directory Server to enforce attribute value uniqueness. For information about the UID Unique Attribute Plugin, see "Working with the UID Unique Attribute Plug-in" in the UnboundID® Directory Proxy Server Administration Guide.

Note: Resource IDs may not be mapped to virtual attributes. For more information about configuring SCIM Resource IDs, see "About the `<resourceIDMapping>` Element".

SCIM Servlet Extension Authentication

Out of the box, the SCIM extension supports basic HTTP authentication with no additional configuration. If the user ID is a valid DN (such as cn=Directory Manager), the SCIM extension authenticates by binding to the Directory 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 directory as that user.

In deployments that use an OAuth authentication server, the SCIM extension can be configured to use OAuth bearer tokens to authenticate. The SCIM extension supports authentication via OAuth 2.0 bearer tokens (per draft-ietf-oauth-v2-bearer-23) using a Server SDK Extension. Because the OAuth 2.0 specification does not specify how contents of a bearer token are formatted, UnboundID provides a token handler API that allows you to handle different implementations of the bearer token from different types of authorization servers.

Basic HTTP authentication is not secure because it passes credentials in clear text. Instead, you need to use HTTP in conjunction with SSL. OAuth also requires SSL, because the bearer token can be sniffed off the network. Therefore, we recommend that you enable SSL on the SCIM Extensions connection handler.

Enabling HTTPS Communications

If you want the SCIM HTTP connection handler to use SSL, which we highly recommend, you need to enable a Key Manager provider and Trust Manager provider.

To enable SSL, include the `--generateSelfSignedCertificate` and the `--ldapsPort` arguments with the `setup` command. If your server already has a certificate that you would like to use, set the `key-manager-provider` to the value you set when you enabled SSL in the Directory Server, or define a new key manager provider. See *Configuring HTTP Connection Handlers*.

To Enable OAuth Authentication

To enable OAuth authentication, you need to create an implementation of the `OAuthTokenHandler` using the API provided in the Server SDK. For details on creating an `OAuthTokenHandler` extension, see the UnboundID Server SDK documentation for more details.

1. Install your OAuth token handler on the server using `dsconfig`.

```
$ bin/dsconfig create-oauth-token-handler \
  --handler-name ExampleOAuthTokenHandler \
  --type third-party \
  --set extension-class:com.unboundid.directory.sdk.examples.ExampleOAuthTokenHandler
```

2. Configure the SCIM servlet extension to use it as follows:

```
$ bin/dsconfig set-http-servlet-extension-prop \
  --extension-name SCIM \
  --set oauth-token-handler:ExampleOAuthTokenHandler
```

Using HTTP Basic Authentication with Bare UID on the Directory Proxy Server

You should be aware that a SCIM server hosted by a Directory Proxy Server rejects HTTP basic authentication requests when a bare UID value is submitted. Unless subordinate base DNs have been defined on the Proxy Server to properly interpret which base DNs are in use in the backend directory servers.

The following example shows that authenticating to the Proxy Server with a bare UID value results in an "Invalid Credentials" error.

```
 $ curl --insecure --include --basic \
  --user "user.1:password" "https://proxy.example.com/Users/
uid=user.1,ou=people,dc=example,dc=com"
HTTP/1.1 401 Unauthorized
WWW-Authenticate: Basic realm=SCIM
Access-Control-Allow-Credentials: true
Content-Type: application/json
Content-Length: 63
Server: Jetty(8.1.3.v20120416)
```

```
{"Errors":[{"code":"401","description":"Invalid credentials"}]}
```

If you use a full DN, then you can properly authenticate to the server.

```
 $ curl --insecure --include --basic \
  --user "uid=user.1,ou=people,dc=example,dc=com:password" "https://proxy.example.com/
Users/uid=user.1,ou=people,dc=example,dc=com"
HTTP/1.1 200 OK
Access-Control-Allow-Credentials: true
Content-Type: application/json
Location: https://proxy.example.com:443/Users/uid=user.1,ou=people,dc=example,dc=com
Content-Length: 1273
Server: Jetty(8.1.3.v20120416)

{"schemas":["urn:scim:schemas:extension:custom:1.0","urn:scim:schemas:core:1.0"
"urn:scim:schemas:extension:enterprise:1.0"],"id":"uid=user.1,ou=people,dc=example,dc=com",
"meta":{"location":"https://proxy.example.com:443/Users/
uid=user.1,ou=people,dc=example,dc=com"}...
```

If you want the ability to use a UID value, and not the full DN, you can tell the Proxy Server to use subordinate base DNs in the backend directory servers. Run the following `dsconfig` command on the Proxy Server, after which you will be able to authenticate to the SCIM Server using UID values.

```
$ bin/dsconfig set-root-dse-backend-prop \
  --set subordinate-base-dn:dc=example,dc=com
```

Note: This step is not necessary if the SCIM server is hosted by a Directory Server as it already knows about the subordinate base DNs.

Managing the SCIM Schema

SCIM provides a common user schema and extension model, making it easier to interoperate with multiple Service Providers. This section describes the SCIM schema and provides information on how to map LDAP schema to the SCIM resource schema.

About SCIM Schema

SCIM provides a common user schema and extension model, making it easier to interoperate with multiple Service Providers. The core SCIM schema defines a concrete schema for user and group resources that encompasses common attributes found in many existing schemas.

Each attribute is defined as either a single attribute, allowing only one instance per resource, or a multi-valued attribute, in which case several instances may be present for each resource. Attributes may be defined as simple, name-value pairs or as complex structures that define sub-attributes.

While the SCIM schema follows an object extension model similar to object classes in LDAP, it does not have an inheritance model. Instead, all extensions are additive, similar to LDAP Auxiliary Object Classes.

Mapping LDAP Schema to SCIM Resource Schema

The resources configuration file is an XML file that is used to define the SCIM resource schema and its mapping to LDAP schema. The default configuration of the `scim-resources.xml` file provides definitions for the standard SCIM Users and Groups resources, and mappings to the standard LDAP `inetOrgPerson` and `groupOfUniqueNames` object classes.

The default configuration may be customized by adding extension attributes to the Users and Groups resources, or by adding new extension resources. The resources file is composed of a single `<resources>` element, containing one or more `<resource>` elements.

For any given SCIM resource endpoint, only one `<LDAPAdd>` template can be defined, and only one `<LDAPSearch>` element can be referenced. If entries of the same object class can be located under different subtrees or base DNs of the Directory Server, then a distinct SCIM resource must be defined for each unique entry location in the directory information tree. This can be implemented in many ways. For example:

- Create multiple SCIM servlets, each with a unique `scim-resources.xml` configuration, and each running under a unique HTTP connection handler.
- Create multiple SCIM servlets, each with a unique `scim-resources.xml` configuration, each running under a single, shared HTTP connection handler, but each with a unique context path.

The remainder of this section describes the mapping elements available in the `scim-resources.xml` file.

About the <resource> Element

A `resource` element has the following XML attributes:

- `schema`: a required attribute specifying the SCIM schema URN for the resource. Standard SCIM resources already have URNs assigned for them, such as `urn:scim:schemas:core:1.0`. A new URN must be obtained for custom resources using any of the standard URN assignment methods.
- `name`: a required attribute specifying the name of the resource used to access it through the SCIM REST API.
- `mapping`: a custom Java class that provides the logic for the resource mapper. This class must extend the `com.unboundid.scim.ldap.ResourceMapper` class.

A resource element contains the following XML elements in sequence:

- `description`: a required element describing the resource.
- `endpoint`: a required element specifying the endpoint to access the resource using the SCIM REST API.

- `LDAPSearchRef`: a mandatory element that points to an `LDAPSearch` element. The `LDAPSearch` element allows a SCIM query for the resource to be handled by an LDAP service and also specifies how the SCIM resource ID is mapped to the LDAP server.
- `LDAPAdd`: an optional element specifying information to allow a new SCIM resource to be added through an LDAP service. If the element is not provided then new resources cannot be created through the SCIM service.
- `attribute`: one or more elements specifying the SCIM attributes for the resource.

About the <attribute> Element

An `attribute` element has the following XML attributes:

- `schema`: a required attribute specifying the schema URN for the SCIM attribute. If omitted, the schema URN is assumed to be the same as that of the enclosing resource, so this only needs to be provided for SCIM extension attributes. Standard SCIM attributes already have URNs assigned for them, such as urn:scim:schemas:core:1.0. A new URN must be obtained for custom SCIM attributes using any of the standard URN assignment methods.
- `name`: a required attribute specifying the name of the SCIM attribute.
- `readOnly`: an optional attribute indicating whether the SCIM sub-attribute is not allowed to be updated by the SCIM service consumer. The default value is false.
- `required`: an optional attribute indicating whether the SCIM attribute is required to be present in the resource. The default value is false.

An attribute element contains the following XML elements in sequence:

- `description`: a required element describing the attribute. Then just one of the following elements:
  - `simple`: specifies a simple, singular SCIM attribute.
  - `complex`: specifies a complex, singular SCIM attribute.
  - `simpleMultiValued`: specifies a simple, multi-valued SCIM attribute.
  - `complexMultiValued`: specifies a complex, multi-valued SCIM attribute.

About the <simple> Element

A `simple` element has the following XML attributes:

- `dataType`: a required attribute specifying the simple data type for the SCIM attribute. The following values are permitted: binary, boolean, dateTime, decimal, integer, string.
- `caseExact`: an optional attribute that is only applicable for string data types. It indicates whether comparisons between two string values use a case-exact match or a case-ignore match. The default value is false.

A simple element contains the following XML element:

- `mapping`: an optional element specifying a mapping between the SCIM attribute and an LDAP attribute. If this element is omitted, then the SCIM attribute has no mapping and the SCIM service ignores any values provided for the SCIM attribute.

About the <complex> Element

The `complex` element does not have any XML attributes. It contains the following XML element:

- `subAttribute`: one or more elements specifying the sub-attributes of the complex SCIM attribute, and an optional mapping to LDAP. The standard `type`, `primary`, and `display` sub-attributes do not need to be specified.

About the <simpleMultivalued> Element

A `simpleMultiValued` element has the following XML attributes:

- `childName`: a required attribute specifying the name of the tag that is used to encode values of the SCIM attribute in XML in the REST API protocol. For example, the tag for the standard emails SCIM attribute is email.
- `dataType`: a required attribute specifying the simple data type for the plural SCIM attribute (i.e. the data type for the value sub-attribute). The following values are permitted: `binary`, `boolean`, `dateTime`, `integer`, `string`.
- `caseExact`: an optional attribute that is only applicable for string data types. It indicates whether comparisons between two string values use a case-exact match or a case-ignore match. The default value is false.

A simpleMultiValued element contains the following XML elements in sequence:

- `canonicalValue`: specifies the values of the type sub-attribute that is used to label each individual value, and an optional mapping to LDAP.
- `mapping`: an optional element specifying a default mapping between the SCIM attribute and an LDAP attribute.

About the <complexMultiValued> Element

A `complexMultiValued` element has the following XML attribute:

- `tag`: a required attribute specifying the name of the tag that is used to encode values of the SCIM attribute in XML in the REST API protocol. For example, the tag for the standard addresses SCIM attribute is address.

A `complexMultiValued` element contains the following XML elements in sequence:

- `subAttribute`: one or more elements specifying the sub-attributes of the complex SCIM attribute. The standard type, primary, and display sub-attributes do not need to be specified.

- `canonicalValue`: specifies the values of the type sub-attribute that is used to label each individual value, and an optional mapping to LDAP.

About the <subAttribute> Element

A `subAttribute` element has the following XML attributes:

- `name`: a required element specifying the name of the sub-attribute.
- `readOnly`: an optional attribute indicating whether the SCIM sub-attribute is not allowed to be updated by the SCIM service consumer. The default value is false.
- `required`: an optional attribute indicating whether the SCIM sub-attribute is required to be present in the SCIM attribute. The default value is false.
- `dataType`: a required attribute specifying the simple data type for the SCIM sub-attribute. The following values are permitted: binary, boolean, dateTime, integer, string.
- `caseExact`: an optional attribute that is only applicable for string data types. It indicates whether comparisons between two string values use a case-exact match or a case-ignore match. The default value is false.

A `subAttribute` element contains the following XML elements in sequence:

- `description`: a required element describing the sub-attribute.
- `mapping`: an optional element specifying a mapping between the SCIM sub-attribute and an LDAP attribute. This element is not applicable within the complexMultiValued element.

About the <canonicalValue> Element

A `canonicalValue` element has the following XML attribute:

- `name`: specifies the value of the type sub-attribute. For example, work is the value for emails, phone numbers and addresses intended for business purposes.

A `canonicalValue` element contains the following XML element:

- `subMapping`: an optional element specifying mappings for one or more of the sub-attributes. Any sub-attributes that have no mappings will be ignored by the mapping service.

About the <mapping> Element

A `mapping` element has the following XML attributes:

- `ldapAttribute`: A required element specifying the name of the LDAP attribute to which the SCIM attribute or sub-attribute map.

- `transform`: An optional element specifying a transformation to apply when mapping an attribute value from SCIM to LDAP and vice-versa. The available transformations are described in *Mapping LDAP Entries to SCIM Using the SCIM-LDAP API*.

About the <subMapping> Element

A `subMapping` element has the following XML attributes:

- `name`: a required element specifying the name of the sub-attribute that is mapped.
- `ldapAttribute`: a required element specifying the name of the LDAP attribute to which the SCIM sub-attribute maps.
- `transform`: an optional element specifying a transformation to apply when mapping an attribute value from SCIM to LDAP and vice-versa. The available transformations are described later. The available transformations are described in *Mapping LDAP Entries to SCIM Using the SCIM-LDAP API*.

About the <LDAPSearch> Element

An `LDAPSearch` element contains the following XML elements in sequence:

- `baseDN`: a required element specifying 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 via an ID attribute.

Note: The `LDAPSearch` element can be added as a top-level element outside of any <Resource> elements, and then referenced within them via an ID attribute.

About the <resourceIDMapping> Element

The `resourceIDMapping` element has the following XML attributes:

- `ldapAttribute`: a required element specifying the name of the LDAP attribute to which the SCIM resource ID maps.
- `createdBy`: a required element specifying the source of the resource ID value when a new resource is created by the SCIM consumer using a POST operation. Allowable values for this element include `scim-consumer`, meaning that a value must be present in the initial resource content provided by the SCIM consumer, or directory, meaning that a value is automatically

provided by the Directory Server (as would be the case if the mapped LDAP attribute is entryUUID).

The following example illustrates an `LDAPSearch` element that contains a `resourceIDMapping` element:

```
<LDAPSearch id="userSearchParams">
  <baseDN>ou=people,dc=example,dc=com</baseDN>
  <filter>(objectClass=inetOrgPerson)</filter>
  <resourceIDMapping ldapAttribute="entryUUID" createdBy="directory"/>
</LDAPSearch>
```

About the <LDAPAdd> Element

An LDAPAdd element contains the following XML elements in sequence:

- `DNTemplate`: a required element specifying a template that is used to construct the DN of an entry representing a SCIM resource when it is created. The template may reference values of the entry after it has been mapped using `{ldapAttr}`, where `ldapAttr` is the name of an LDAP attribute.
- `fixedAttribute`: zero or more elements specifying fixed LDAP values to be inserted into the entry after it has been mapped from the SCIM resource.

About the <fixedAttribute> Element

A `fixedAttribute` element has the following XML attributes:

- `ldapAttribute`: a required attribute specifying the name of the LDAP attribute for the fixed values.
- `onConflict`: an optional attribute specifying the behavior when the LDAP entry already contains the specified LDAP attribute. The value merge indicates that the fixed values should be merged with the existing values. The value overwrite indicates that the existing values are to be overwritten by the fixed values. The value preserve indicates that no changes should be made. The default value is merge.

A `fixedAttribute` element contains one or more `fixedValue` XML element, which specify the fixed LDAP values.

Validating Updated SCIM Schema

The UnboundID 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
```

Using Pre-defined Transformations

The following pre-defined transformations may be referenced by the transform XML attribute:

- `com.unboundid.scim.ldap.BooleanTransformation` Transforms SCIM boolean data type values to LDAP Boolean syntax values and vice-versa.
- `com.unboundid.scim.ldap.GeneralizedTimeTransformation` Transforms SCIM dateTime data type values to LDAP Generalized Time syntax values and vice-versa.
- `com.unboundid.scim.ldap.PostalAddressTransformation` Transforms SCIM formatted address values to LDAP Postal Address syntax values and vice-versa. SCIM formatted physical mailing addresses are represented as strings with embedded new lines, whereas LDAP uses the $ character to separate address lines. This transformation interprets new lines in SCIM values as address line separators.

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, UnboundID provides a library called SCIM-LDAP, which provides facilities for writing custom transformations and more advanced mapping. This API is provided with the SCIM Reference Implementation. It is also available via the Maven Central public repository at: http://search.maven.org.

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>1.2.0</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 extension's lib directory.

Testing SCIM Query Performance

You can use the `scim-query-rate` tool, provided in the SCIM Reference Implementation, to test query performance, by performing repeated resource queries against the SCIM server. You should be aware that this tool is bundled with the SCIM Reference Implementation, and not the SCIM extension. The source code for the SCIM Reference Implementation is available from Google code at: http://scimsdk.googlecode.com.

The `scim-query-rate` tool performs searches using a query filter or can request resources by ID. For example, you can test performance by using a filter to query randomly across a set of one million users with eight concurrent threads. The user resources returned to the client in this example is in XML format and includes the `userName` and `name` attributes.

```
scim-query-rate --hostname server.example.com --port 80 \
--authID admin --authPassword password --xml \
--filter 'userName eq "user.[1-1000000]"' --attribute userName \
--attribute name --numThreads 8
```

You can request resources by specifying a resource ID pattern using the `--resourceID` argument as follows:

```
scim-query-rate --hostname server.example.com --port 443 \
--authID admin --authPassword password --useSSL --trustAll\
--resourceName User \
--resourceID 'uid=user.[1-150000],ou=people,dc=example,dc=com'
```

The `scim-query-rate` tool reports the error `"java.net.SocketException: Too many open files"` if the open file limit is too low. You can increase the open file limit to increase the number of file descriptors.

Monitoring Resources Using the SCIM Extension

The monitor provider exposes the following information for each resource:

- Number of successful operations per request type (such as GET, PUT, and POST).
- Number of failed operations and their error codes per request type.
- Number of operations with XML or JSON from client.
- Number of operations that sent XML or JSON to client.

In addition to the information about the user-defined resources, monitoring information is also generated for the schema, service provider configuration, and monitor resources. The attributes of the monitor entry are formatted as follows:

```
{resource name}-resource-{request type}-{successful or error status code}
```

You can search for one of these monitor providers using an ldapsearch such as the following:

```
$ bin/ldapsearch --port 1389 bindDN uid=admin,dc=example,dc=com \
  --bindPassword password --baseDN cn=monitor \
  --searchScope sub "(objectclass=scim-servlet-monitor-entry)"
```

For example, the following monitor output was produced by a test environment with three distinct SCIM servlet instances, Aleph, Beth, and Gimel. Note that the first instance has a custom resource type called host.

```
$ bin/ldapsearch --baseDN cn=monitor \
  '(objectClass=scim-servlet-monitor-entry)'
dn: cn=SCIM Servlet (SCIM HTTP Connection Handler),cn=monitor
objectClass: top
objectClass: ds-monitor-entry
objectClass: scim-servlet-monitor-entry
objectClass: extensibleObject
cn: SCIM Servlet (SCIM HTTPS Connection Handler) [from
  ThirdPartyHTTPServletExtension:SCIM (Aleph)]
ds-extension-monitor-name: SCIM Servlet (SCIM HTTPS Connection Handler)
ds-extension-type: ThirdPartyHTTPServletExtension
ds-extension-name: SCIM (Aleph)
version: 1.2.0
build: 20120105174457Z
revision: 820
schema-resource-query-successful: 8
schema-resource-query-401: 8
schema-resource-query-response-json: 16
user-resource-delete-successful: 1
user-resource-put-content-xml: 27
```

```
user-resource-query-response-json: 3229836
user-resource-put-403: 5
user-resource-put-content-json: 2
user-resource-get-401: 1
user-resource-put-response-json: 23
user-resource-get-response-json: 5
user-resource-get-response-xml: 7
user-resource-put-400: 2
user-resource-query-401: 1141028
user-resource-post-content-json: 1
user-resource-put-successful: 22
user-resource-post-successful: 1
user-resource-delete-404: 1
user-resource-query-successful: 2088808
user-resource-get-successful: 10
user-resource-put-response-xml: 6
user-resource-get-404: 1
user-resource-delete-401: 1
user-resource-post-response-json: 1
host-resource-query-successful: 5773268
host-resource-query-response-json: 11576313
host-resource-query-400: 3
host-resource-query-response-xml: 5
host-resource-query-401: 5788152
dn: cn=SCIM Servlet (SCIM HTTP Connection Handler),cn=monitor
objectClass: top
objectClass: ds-monitor-entry
objectClass: scim-servlet-monitor-entry
objectClass: extensibleObject
cn: SCIM Servlet (SCIM HTTPS Connection Handler) [from
  ThirdPartyHTTPServletExtension:SCIM (Beth)]
ds-extension-monitor-name: SCIM Servlet (SCIM HTTPS Connection
  Handler)
ds-extension-type: ThirdPartyHTTPServletExtension
ds-extension-name: SCIM (Beth)
version: 1.2.0
build: 20120105174457Z
revision: 820
serviceproviderconfig-resource-get-successful: 3
serviceproviderconfig-resource-get-response-json: 2
serviceproviderconfig-resource-get-response-xml: 1
schema-resource-query-successful: 8
schema-resource-query-401: 8
schema-resource-query-response-json: 16
group-resource-query-successful: 245214
group-resource-query-response-json: 517841
group-resource-query-400: 13711
group-resource-query-401: 258916
user-resource-query-response-json: 107876
user-resource-query-400: 8288
user-resource-get-400: 33
user-resource-get-response-json: 1041
user-resource-get-successful: 2011
user-resource-query-successful: 45650
user-resource-get-response-xml: 1003
user-resource-query-401: 53938
dn: cn=SCIM Servlet (SCIM HTTP Connection Handler),cn=monitor
objectClass: top
objectClass: ds-monitor-entry
objectClass: scim-servlet-monitor-entry
objectClass: extensibleObject
cn: SCIM Servlet (SCIM HTTPS Connection Handler) [from
  ThirdPartyHTTPServletExtension:SCIM (Gimel)]
ds-extension-monitor-name: SCIM Servlet (SCIM HTTPS Connection
  Handler)
ds-extension-type: ThirdPartyHTTPServletExtension
ds-extension-name: SCIM (Gimel)
version: 1.2.0
build: 20120105174457Z
revision: 820
schema-resource-query-successful: 1
schema-resource-query-401: 1
schema-resource-query-response-json: 2
user-resource-query-successful: 65
user-resource-get-successful: 4
user-resource-get-response-json: 6
user-resource-query-response-json: 132
user-resource-get-404: 2
user-resource-query-401: 67
```

Monitoring Internal SCIM Servlet Extension Operations

The SCIM extension rewrites incoming HTTP requests as internal LDAP operations. You can create a logger for these internal operations to troubleshoot a problem or to tune the underlying Directory Server.

For example, you can create a request criteria object that will be used to match as closely as possible any operations initiated by the SCIM extension. This example assumes that all user and group entries use the base DNs `"dc=example,dc=com" and "dc=example,dc=org"`.

```
$ bin/dsconfig create-request-criteria \
  --criteria-name "Example Co Internal Operations Request Criteria" \
  --type simple \
  --set operation-origin:internal-operation \
  --set included-target-entry-dn:dc=example,dc=com \
  --set included-target-entry-dn:dc=example,dc=org \
  --set using-administrative-session-worker-thread:false
```

Using `set operation-origin:internal-operation` ensures that external requests from LDAP clients are not matched. Because the SCIM extension does not use administrative sessions, we set the `using-administrative-session-worker-thread` attribute to `false`. The `included-target-entry-dn` values are used to filter out requests against administrative and monitoring backends, such as `cn=config` or `cn=monitor`.

Next, create a log publisher that uses the above request criteria object.

```
$ bin/dsconfig create-log-publisher \
  --publisher-name "Internal Operations Access Logger" \
  --type file-based-access --set enabled:true \
  --set suppress-internal-operations:false \
  --set suppress-replication-operations:true \
  --set log-connects:true --set log-disconnects:true \
  --set log-requests:true --set log-search-entries:true \
  --set "request-criteria:Example Co Internal Operations Request Criteria" \
  --set include-request-controls:true \
  --set log-file:logs/internal-ops \
  --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"
```

Note: You may not be able to completely filter out requests generated by sources other than the SCIM servlet extension. This logger picks up requests initiated by other extensions if they use internal resources.

Creating Your Own SCIM Application

SCIM is an open initiative designed to make moving identity data to, from, and between clouds standard, secure, fast, and easy. UnboundID provides an open source SCIM SDK and Reference Implementation with which you can build a SCIM application.

The UnboundID SCIM Reference Implementation is an easy-to-use, self-contained implementation of a SCIM Service Provider (server) and consumer (client). The server is built

on the UnboundID In-Memory Directory Server and allows for custom mappings between LDAP and SCIM data models.

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 reference implementation uses the UnboundID SCIM SDK, UnboundID LDAP SDK for Java, and other open source libraries.

The reference implementation supports all required aspects of the SCIM API, schema, and schema extension model. Both the Reference Implementation and the SCIM SDK are open source and freely distributable under the terms of the GPLv2, LGPLv2.1, and UnboundID Free License.

The SCIM SDK is available for download at the following sites:

- UnboundID repository: http://www.unboundid.com/labs/projects/simple-cloud-identity-management-scim/
- Maven Central public repository: http://search.maven.org

You can use the following dependency element in your project's POM file:

```
<dependency>
   <groupId>com.unboundid.product.scim</groupId>
   <artifactId>scim-sdk</artifactId>
   <version>1.2.0</version>
</dependency>
```

The SCIM Reference Implementation is available for download at: http://www.unboundid.com/labs/projects/simple-cloud-identity-management-scim/.

The source code for the SCIM Reference Implementation is available from Google Code at: http://scimsdk.googlecode.com.